MIT CSIL • COMPUTER SCIENCE EDUCATION

2019 EDITION

The Missing Semester of Your CS Education

Burmese

ORIGINAL COURSE

Anish Athalye, Jon Gjengset, & Jose Javier Gonzalez Ortiz

Massachusetts Institute of Technology (MIT)

Original Website: https://missing.csail.mit.edu/

MY LOCALIZATION & PUBLICATION

FOSS Myanmar (fossmyanmar.org) & **Vibe Code Tours** (vibecode.tours)

Burmese Website: https://missing-semester-my.github.io/

မာတိကာ (Table of Contents)

- Virtual environment များ
- Vendoring
- Program Introspection
- Debugging
  - GDB/LLDB
  - PDB
  - Web browser Developer Tools
- strace
- Profiling
  - Go
  - Perf
- Remote Machines
  - Command များကို မောင်းနှင်ခြင်း
  - SSH Key များ
  - SSH မှတစ်ဆင့် ဖိုင်များကို ကူးယူခြင်း
  - Process များကို Background တွင် လည်ပတ်စေခြင်း
  - Port Forwarding
  - Graphics Forwarding
  - Roaming
  - SSH Configuration
  - Remote Filesystem (အဝေးထိန်း ဖိုင်စနစ်)
  - လေ့ကျင့်ခန်းများ
  - ကိုးကားချက်များ
- Security and Privacy
  - သင့်တော်သော လူများကို Follow လုပ်ပါ (Follow the Right People)
  - ယေဘုယျ လုံခြုံရေး အကြံပြုချက်များ (General Security Advice)

သင်တန်း မိတ်ဆက် (Introduction)

ကွန်ပျူတာသိပ္ပံ (CS) သင်တန်းများတွင် OS များ (Operating Systems) မှစ၍ စက်သင်ယူမှု (Machine Learning) အထိ အဆင့်မြင့် ခေါင်းစဉ်များကို သင်ကြားပေးကြသော်လည်း၊ ကျောင်းသားများ ကိုယ်တိုင် ကြုံစားလေ့လာယူရလေ့ရှိပြီး ပုံမှန် သင်ရိုးများတွင် ထည့်သွင်းသင်ကြားပေးလေ့မရှိသော အရေးကြီးသည့် ဘာသာရပ်တစ်ခု ရှိနေပါသည်—ယင်းမှာ မိမိတို့ အသုံးပြုသည့် tool များကို ကျွမ်းကျင်စွာ ကိုင်တွယ်နိုင်မှုပင် ဖြစ်သည်။ ဤသင်တန်းတွင် Command-line ကို ကျွမ်းကျင်စွာ အသုံးပြုပုံ၊ စွမ်းအားထက်မြက်သော Text Editor များ အသုံးပြုပုံ၊ Version Control စနစ်များ၏ အဆင့်မြင့် လုပ်ဆောင်ချက်များ အသုံးပြုပုံနှင့် အခြား အရေးပါသော အကြောင်းအရာများကို သင်ကြားပေးသွားမည် ဖြစ်ပါသည်။

ကျောင်းသားများသည် မိမိတို့၏ ကျောင်းသားဘဝတွင် ဤ tool များကို အသုံးပြု၍ နာရီပေါင်း ရာချီ (နှင့် သက်မွေးဝမ်းကျောင်း လုပ်ငန်းခွင်တွင် နာရီပေါင်း သောင်းချီ) အချိန်ကုန်လွန်ကြရသဖြင့်၊ ယင်း tool များကို တတ်နိုင်သမျှ ချောမွေ့လွယ်ကူစွာ အသုံးပြုနိုင်အောင် ပြုလုပ်ထားခြင်းသည် အလွန်အကျိုးရှိလှပါသည်။ ဤ tool များကို ကျွမ်းကျင်အောင် သင်ယူခြင်းဖြင့် tool များနှင့် ရုန်းကန်ရသည့် အချိန်များကို လျှော့ချပေးနိုင်ရုံမျှ မက၊ ယခင်က အလွန်ခက်ခဲနက်နဲသည်ဟု ထင်မှတ်ခဲ့သော ပြဿနာများကိုပါ ဖြေရှင်းနိုင်စွမ်း ရရှိစေမည် ဖြစ်ပါသည်။

ယနေ့ခေတ်တွင် AI နည်းပညာသုံး tool များနှင့် လုပ်ငန်းစဉ်များ ပေါ်ထွက်လာခြင်းကြောင့် ဆော့ဖ်ဝဲလ် အင်ဂျင်နီယာ ဘာသာရပ်၏ အသွင်အပြင်များစွာ ပြောင်းလဲလျက် ရှိပါသည်။ ယင်း tool များကို အကန့်အသတ်များနှင့်တကွ သင့်လျော်စွာ အသုံးပြုတတ်ပါက CS ကျွမ်းကျင်သူများအတွက် များစွာ အကျိုး ရှိစေမည် ဖြစ်၍ လက်တွေ့ အသုံးပြုနိုင်သည့် အသိပညာ ရရှိထားရန် လိုအပ်ပါသည်။ AI သည် နယ်ပယ် ပေါင်းစုံတွင် အထောက်အကူပြုသော နည်းပညာဖြစ်သဖြင့် သီးခြား AI သင်ခန်းစာအဖြစ် မထားရှိဘဲ၊ နောက်ဆုံးပေါ် AI tool များနှင့် နည်းလမ်းများကို သက်ဆိုင်ရာ သင်ခန်းစာ တစ်ခုချင်းစီတွင် တိုက်ရိုက် ထည့်သွင်းပေးထားပါသည်။

ဤသင်တန်းကို ဖန်တီးခဲ့သည့် ရည်ရွယ်ချက်ကို ဖတ်ရှုပါ (Motivation) • 📚 အီးဘွတ်ခ် စာအုပ်များ ဒေါင်းလုဒ်ဆွဲရန် (eBook & PDF)။

သင်ရိုးညွှန်းတမ်း (Syllabus)

- **Automation**
- **Backups**
- **Command-line ပတ်ဝန်းကျင်**
- **Course Overview**
- **ဒေတာ ပြုပြင်စီမံခြင်း (Data Wrangling)**
- **Dotfiles**
- **Editors**
- **2019 Lectures**
- **Machine Introspection**
- **OS Customization**
- **Package Management နှင့် Dependency Management**
- **Program Introspection**
- **Remote Machines**
- **Security and Privacy**
- **Shell နှင့် Script ရေးသားခြင်း**
- **Version Control**
- **Virtual Machines နှင့် Containers များ**
- **Web နှင့် Web Browser များ**

လွန်ခဲ့သော နှစ်များမှ အထူး ခေါင်းစဉ်များ (Special topics from previous years)

ကျွန်ုပ်တို့ သင်ကြားပေးသော ခေါင်းစဉ်များသည် တစ်နှစ်နှင့်တစ်နှစ် ကွဲပြားနိုင်ပါသည်။ လွန်ခဲ့သော နှစ်များတွင် သင်ကြားခဲ့သော်လည်း ၂၀၂၆ တွင် မပါဝင်သေးသော ခေါင်းစဉ်များကို လေ့လာလိုသူများအတွက် အောက်ပါအတိုင်း စုစည်းဖော်ပြပေးထားပါသည်။

- *No special topics listed.*

အထွေထွေ အချက်အလက်များ (General information)

သင်ကြားပေးသူများ: ဤသင်တန်းကို Anish၊ Jon နှင့် Jose တို့မှ ပူးတွဲ သင်ကြားပေးပါသည်။
မေးမြန်းရန်: missing-semester@mit.edu သို့ အီးမေးလ်ဖြင့် မေးမြန်းနိုင်ပါသည်။
ဆွေးနွေးပွဲများ: OSSU Discord (မေးခွန်းများအတွက် `#missing-semester-forum` နှင့် တိုက်ရိုက် ဆွေးနွေးရန် `#missing-semester` ကို အသုံးပြုပါ)။

Beyond MIT (MIT အပြင်ဘက် လက်တွေ့ကမ္ဘာ)

Beyond MIT (MIT အပြင်ဘက် လက်တွေ့ကမ္ဘာ) မှ လေ့လာသူများလည်း ဤ အရင်းအမြစ်များမှ အကျိုးကျေးဇူး ရရှိနိုင်ရန်အတွက် မျှဝေပေးထားပါသည်။ အောက်ပါ link များတွင် ဆွေးနွေးချက်များကို ဝင်ရောက် ကြည့်ရှုနိုင်ပါသည်-

- Hacker News (2026, 2020, 2019)
- Lobsters (2026, 2020, 2019)
- r/learnprogramming (2026, 2020, 2019)
- r/programming (2020, 2019)
- X (2026, 2020, 2019)
- Bluesky (2026)
- Mastodon (2026)
- LinkedIn (2026)
- YouTube (2026, 2020, 2019)

ဘာသာပြန်ဆိုချက်များ (Translations)

- Arabic
- Bengali
- Burmese
- Chinese (Simplified)
- Chinese (Traditional, Taiwan)
- German
- Italian
- Japanese
- Kannada
- Korean
- Mongolian
- Persian
- Portuguese
- Russian
- Serbian
- Spanish
- Swedish
- Thai
- Turkish
- Vietnamese

Note: ဤသည်တို့မှာ အသိုင်းအဝိုင်းမှ ပြုလုပ်ထားသော ပြင်ပ ဘာသာပြန် link များ ဖြစ်ကြပြီး၊ ကျွန်ုပ်တို့သည် ၎င်းတို့ကို စစ်ဆေးအတည်ပြုထားခြင်း မရှိပါ။

သင်သည် ဤသင်တန်း၏ သင်ခန်းစာများကို အခြားဘာသာသို့ ဘာသာပြန်ဆိုထားပါက ကျွန်ုပ်တို့၏ စာရင်းတွင် ထည့်သွင်းနိုင်ရန် pull request တင်သွင်းနိုင်ပါသည်။

ကျေးဇူးတင်လွှာ (Acknowledgments)

သင်ခန်းစာ ဗီဒီယိုများ ရိုက်ကူးနိုင်ရန် ကူညီပေးခဲ့ကြသော Elaine Mello နှင့် **MIT Open Learning** တို့ကို ကျေးဇူးတင်ရှိပါသည်။ **SIPB IAP 2026** ၏ အစိတ်အပိုင်းတစ်ခုအဖြစ် ဤသင်တန်းကို ကူညီပံ့ပိုးပေးခဲ့သော Luis Turino / **SIPB** တို့ကိုလည်း အထူး ကျေးဇူးတင်ရှိပါသည်။

Source code.

Source code - MY.

See **here** for contribution & translation guidelines.

ဤသင်တန်းကို သင်ကြားပေးရသည့် ရည်ရွယ်ချက် (Course Motivation)

ပုံမှန် ကွန်ပျူတာသိပ္ပံ ပညာရေးတွင် OS များ (Operating Systems) မှစ၍ ပရိုဂရမ်းမင်း ဘာသာစကားများ နှင့် စက်သင်ယူမှု (Machine Learning) အထိ CS ၏ အဆင့်မြင့် ခေါင်းစဉ်များစွာကို သင်ကြားရလေ့ ရှိပါသည်။ သို့သော် အဖွဲ့အစည်း အများစုတွင် ထည့်သွင်းသင်ကြားပေးလေ့မရှိဘဲ ကျောင်းသားများ ကိုယ်တိုင် လေ့လာယူရန် ချန်ထားလေ့ရှိသော အရေးကြီးသည့် ခေါင်းစဉ်တစ်ခု ရှိနေပါသည်—ယင်းမှာ ကွန်ပျူတာ နည်းပညာ ဂေဟစနစ်နှင့် tool များကို ကျွမ်းကျင်စွာ အသုံးပြုနိုင်မှုပင် ဖြစ်သည်။

နှစ်များစွာအတွင်း MIT တွင် သင်တန်းများစွာ ပူးတွဲ သင်ကြားပေးရင်း ကျောင်းသားများစွာသည် မိမိတို့ အသုံးပြုနိုင်သည့် tool များအကြောင်း ဗဟုသုတ နည်းပါးနေသည်ကို မကြာခဏ တွေ့မြင်ခဲ့ရသည်။ ကွန်ပျူတာများကို လက်ဖြင့် ပြုလုပ်ရသည့် အလုပ်များကို အလိုအလျောက် ပြုလုပ်ရန် ဖန်တီးထားခြင်း ဖြစ်သော်လည်း၊ ကျောင်းသားများသည် ထပ်ခါတလဲလဲ လုပ်ဆောင်ရသည့် အလုပ်များကို လက်ဖြင့်ပင် ပြုလုပ်နေကြဆဲ ဖြစ်ပြီး Version Control နှင့် Text Editor ကဲ့သို့သော စွမ်းအားထက်မြက်သည့် tool များကို အပြည့်အဝ အသုံးမချနိုင်ကြပါ။ အကောင်းဆုံး အခြေအနေတွင် အချိန်ကုန်ပြီး အလုပ်မတွင်ဘဲ ဖြစ်ရသကဲ့သို့၊ အဆိုးဆုံး အခြေအနေတွင် ဒေတာ ဆုံးရှုံးခြင်း သို့မဟုတ် အချို့သော အလုပ်များကို မပြီးမြောက်နိုင်ခြင်း အထိ ဖြစ်စေနိုင်ပါသည်။

ဤခေါင်းစဉ်များကို တက္ကသိုလ် သင်ရိုးညွှန်းတမ်းများတွင် ထည့်သွင်းသင်ကြားပေးလေ့မရှိပါ—ကျောင်းသားများအား ဤ tool များကို မည်သို့ အသုံးပြုရမည်၊ သို့မဟုတ် ထိရောက်စွာ အသုံးပြုနည်းကို ပြသပေးခြင်း မရှိသဖြင့် ရိုးရှင်းသင့်သော အလုပ်များတွင် အချိန်နှင့် အားထုတ်မှုများ အဟောသိကံဖွာ ဖြစ်ရပါသည်။ ပုံမှန် CS သင်ရိုးတွင် ကျောင်းသားများ၏ ဘဝကို အလွန် လွယ်ကူ အဆင်ပြေစေနိုင်မည့် ကွန်ပျူတာ နည်းပညာ ဂေဟစနစ်ဆိုင်ရာ အရေးကြီးသော ခေါင်းစဉ်များ လိုအပ်နေပါသည်။

The Missing Semester of Your CS Education

ဤလိုအပ်ချက်ကို ဖြည့်ဆည်းရန်အတွက် ထိရောက်သော ကွန်ပျူတာသိပ္ပံပညာရှင်နှင့် Programmer တစ်ယောက် ဖြစ်လာစေရန် မရှိမဖြစ် လိုအပ်သည်ဟု ကျွန်ုပ်တို့ ယူဆသော ခေါင်းစဉ်များအားလုံး ပါဝင်သည့် သင်တန်းတစ်ခုကို ဖန်တီးခဲ့ပါသည်။ ဤသင်တန်းသည် လက်တွေ့ကျပြီး သင်ကြုံတွေ့ရမည့် အခြေအနေအမျိုးမျိုးတွင် ချက်ချင်း အသုံးချနိုင်မည့် tool များနှင့် နည်းလမ်းများကို လက်တွေ့ မိတ်ဆက်ပေးသွားမည် ဖြစ်ပါသည်။

ဤသင်တန်းတွင် သင်ယူရမည့် အကြောင်းအရာများ၏ လက်တွေ့ စံနမူနာများမှာ-

Command shell

Aliases၊ Scripts နှင့် Build Systems များကို အသုံးပြု၍ ပုံမှန် ထပ်ခါတလဲလဲ ပြုလုပ်ရသော အလုပ်များကို မည်သို့ အလိုအလျောက် (Automate) ပြုလုပ်မည်နည်း။ Text document မှ command များကို ကူးယူ ကူးထည့် (copy-paste) ရသည့် ဒုက္ခများ၊ command ၁၅ ခုကို တစ်ခုပြီးတစ်ခု အစဉ်လိုက် ရိုက်ထည့်ရခြင်းများ၊ argument များ ပို့ရန် မေ့လျော့ခြင်းများ နောက်ထပ် ကြုံတွေ့ရတော့မည် မဟုတ်ပါ။

ဥပမာအားဖြင့်၊ Shell History အတွင်း မြန်ဆန်စွာ ရှာဖွေနိုင်ခြင်းသည် အချိန်များစွာ အသက်သာစေပါသည်။ အောက်ပါ စံနမူနာတွင် `convert` command များအတွက် shell history ကို ရှာဖွေအသုံးပြုပုံ အကွက်ဆန်း အချို့ကို ဖော်ပြထားပါသည်-

📹 **Video Demo (History)**

Scan QR code or visit link to watch demo:

https://missing-semester-my.github.io/static/media/demos/history.mp4

Version control (Git)

Version control ကို စနစ်တကျ မှန်ကန်စွာ အသုံးပြုနည်း၊ ယင်းကို အသုံးချ၍ မတော်တဆ ပျက်စီးမှုများမှ ကာကွယ်နည်း၊ အခြားသူများနှင့် ပူးပေါင်းဆောင်ရွက်နည်း၊ နှင့် ဒုက္ခပေးနေသော အပြောင်းအလဲများကို လျင်မြန်စွာ ရှာဖွေ ခွဲထုတ်နည်း။ `rm -rf; git clone` ပြုလုပ်စရာ မလိုတော့ပါ။ Merge conflict များလည်း (အနည်းဆုံးတော့ ပိုမို နည်းပါးသွားမည်) ဖြစ်ပါသည်။ Comment ပိတ်ထားသော စာကြောင်းကြီးများ ချန်ထားစရာ မလိုတော့ပါ။ ကုဒ် ပျက်စီးသွားသည့် နေရာကို ရှာဖွေရန် စိုးရိမ်စရာ မလိုတော့ပါ။ ဤသင်တန်း

တွင် pull request များ အသုံးပြု၍ အခြားသူများ၏ project များတွင် မည်သို့ ပါဝင်ကူညီရမည်ကိုပါ သင်ကြားပေးသွားမည် ဖြစ်ပါသည်။

အောက်ပါ ဥပမာတွင် `git bisect` ကို အသုံးပြု၍ Unit Test ကို ပျက်စီးစေခဲ့သော commit ကို ရှာဖွေပြီး `git revert` ဖြင့် ပြန်လည် ပြင်ဆင်ပုံကို ဖော်ပြထားပါသည်-

📹 **Video Demo (Git)**

Scan QR code or visit link to watch demo:

https://missing-semester-my.github.io/static/media/demos/git.mp4

Text editing (Vim)

Local နှင့် Remote စက်များရှိ file များကို command-line မှနေ၍ ထိရောက်စွာ Edit ပြုလုပ်နည်းနှင့် အဆင့်မြင့် Editor များ၏ လုပ်ဆောင်ချက်များကို အသုံးချနည်း။ File များကို ရှေ့နောက် ကူးယူနေစရာ မလိုတော့ပါ။ ထပ်ခါတလဲလဲ file ပြင်ဆင်နေစရာ မလိုတော့ပါ။

Vim macros များသည် Vim ၏ အကောင်းဆုံး လုပ်ဆောင်ချက်တစ်ခု ဖြစ်ပြီး အောက်ပါ ဥပမာတွင် HTML table တစ်ခုကို CSV format သို့ Vim macro အသုံးပြု၍ လျင်မြန်စွာ ပြောင်းလဲပုံကို ဖော်ပြထားပါသည်-

📹 **Video Demo (Vim)**

Scan QR code or visit link to watch demo:

https://missing-semester-my.github.io/static/media/demos/vim.mp4

Remote machines (SSH & Terminal Multiplexing)

SSH keys များနှင့် terminal multiplexing များကို အသုံးပြု၍ remote machine များနှင့် အလုပ်လုပ်ရာတွင် စနစ်တကျ အဆင်ပြေစေရန် ပြုလုပ်နည်း။ Command နှစ်ခုကို ပြိုင်တူ လွှတ်ရန်အတွက် Terminal တွေ အများကြီး ဖွင့်ထားစရာ မလိုတော့ပါ။ ချိတ်ဆက်တိုင်း password ပြန်ရိုက်နေစရာ မလိုတော့ပါ။ အင်တာနက် လိုင်းကျသွားခြင်း သို့မဟုတ် Laptop reboot လုပ်လိုက်ရခြင်းကြောင့် အရာအားလုံး ဆုံးရှုံးသွားရခြင်းမျိုး မရှိတော့ပါ။

အောက်ပါ ဥပမာတွင် `tmux` ကို အသုံးပြု၍ remote server များပေါ်တွင် session များကို အသက်ရှင်လျက် ထိန်းသိမ်းထားပုံနှင့် `mosh` အသုံးပြု၍ network ပြောင်းလဲမှုကို ထိန်းသိမ်းပေးပုံကို ဖော်ပြထားပါသည်-

📹 **Video Demo (Ssh)**

Scan QR code or visit link to watch demo:

https://missing-semester-my.github.io/static/media/demos/ssh.mp4

Finding files (ဖိုင်များ ရှာဖွေခြင်း)

သင် ရှာဖွေနေသော file များကို လျင်မြန်စွာ ရှာဖွေနည်း။ မိမိ လိုချင်သော ကုဒ် ပါသည့် file တွေ့သည့်အထိ project ထဲက file တစ်ခုချင်းစီလိုက် နှိပ်ကြည့်နေစရာ မလိုတော့ပါ။

အောက်ပါ ဥပမာတွင် `fd` ဖြင့် file များကို မြန်ဆန်စွာ ရှာဖွေပြီး `rg` ဖြင့် ကုဒ် အစိတ်အပိုင်းများကို ရှာဖွေပုံ၊ ထို့ပြင် `fasd` ဖြင့် မကြာသေးမီက အသုံးပြုခဲ့သော file/folder သို့ လျင်မြန်စွာ `cd` နှင့် `vim` ဝင်ရောက်ပုံတို့ကို ဖော်ပြထားပါသည်-

📹 **Video Demo (Find)**

Scan QR code or visit link to watch demo:

https://missing-semester-my.github.io/static/media/demos/find.mp4

ဒေတာ ပြုပြင်စီမံခြင်း (Data Wrangling)

Command-line မှ တိုက်ရိုက် ဒေတာနှင့် ဖိုင်များကို လျင်မြန် လွယ်ကူစွာ ပြုပြင်ခြင်း၊ ကြည့်ရှုခြင်း၊ Parse လုပ်ခြင်း၊ Plot ဆွဲခြင်းနှင့် တွက်ချက်ခြင်းများ ပြုလုပ်နည်း။ Log file များထဲမှ ကူးယူ ကူးထည့် စရာ မလိုတော့ပါ။ ဒေတာ စာရင်းအင်းများကို လက်ဖြင့် တွက်ချက်နေစရာ မလိုတော့ပါ။

Code quality and continuous integration

Autoformatting၊ Linting၊ Testing နှင့် Code coverage tool များကို အသုံးပြု၍ ကုဒ် အရည်အသွေး တိုးတက်အောင် ပြုလုပ်နည်း။ ရုပ်ဆိုးသော ကုဒ်များ၊ မလိုလားအပ်သော အမှားဟောင်းများ ပြန်ဖြစ်ခြင်းများ၊ မိမိစက်တွင် အလုပ်လုပ်ပြီး အခြားသူ စက်တွင် ပျက်စီးသွားသော ကုဒ်များ မရှိတော့ပါ။

Beyond the code (ကုဒ်ရေးသားခြင်းထက် ကျော်လွန်၍)

ကောင်းမွန်သော Documentation ရေးသားနည်း၊ Open-source ထိန်းသိမ်းသူများနှင့် ရှင်းလင်းစွာ ဆက်သွယ်နည်း၊ တိကျသော Issue များ တင်သွင်းနည်းနှင့် Merge လုပ်ခံရမည့် Pull Request များ ပါဝင်ကူညီနည်း။

နိဂုံး (Conclusion)

ဤအကြောင်းအရာများနှင့် အခြား အကြောင်းအရာများစွာကို သင်ခန်းစာများအတွင်း သင်ကြားပေးသွားမည် ဖြစ်ပါသည်။ ဇန်နဝါရီ မတိုင်မီ စောစီးစွာ လေ့လာလိုပါက ယခင် သင်တန်း **2020 သင်ခန်းစာများ** ကိုလည်း ဝင်ရောက် ကြည့်ရှုနိုင်ပါသည်။

Happy hacking,

Anish၊ **Jon** နှင့် **Jose**

Course Overview

▶ LECTURE VIDEO REFERENCE

Course Overview

Scan the QR code below using your mobile camera or click the link to watch this full lecture online:

Direct Online Link:

https://www.youtube.com/watch?v=qw2c6ffSVOM

စိတ်ဓာတ်တက်ကြွမှုနှင့် ရည်ရွယ်ချက် (Motivation)

ဤသင်တန်းသည် ဖန်တီးတီထွင်သူများ၏ **hacker culture** (https://en.wikipedia.org/wiki/Hacker_culture) tool များ အကြောင်း သင်ကြားပေးခြင်း ဖြစ်ပြီး လုံခြုံရေးဖောက်ထွင်းသည့် **security hacker** (https://en.wikipedia.org/wiki/Security_hacker) tool များအကြောင်း မဟုတ်ပါ။

MIT သင်တန်းများတွင် ဤအကြောင်းအရာများကို အသေးစိတ် သင်ကြားပေးခြင်း မရှိပါ။ မိမိ၏ tool များကို ကွန်မန်းကျင်စွာ အသုံးပြုနိုင်ခြင်းသည် အလွန်ပင် အကျိုးကျေးဇူး ကြီးမားလှသည်—ယင်းက သင်၏ အချိန်များစွာကို သက်သာစေမည် ဖြစ်သည် (အကျိုးအမြတ် ပြန်ရသည့် အချိန်မှာလည်း အလွန် တိုတောင်းပါသည်)။

ကွန်ပ်တျူသည် tool အသစ်များအကြောင်း၊ မိမိ၏ tool များကို အထိရောက်ဆုံး မည်သို့ အသုံးချရမည်၊ မိမိ၏ tool များကို စိတ်ကြိုက် မည်သို့ စီမံပြင်ဆင်ရမည် (customize) နှင့် tool များကို ပိုမို တိုးချဲ့ အသုံးပြုနည်း (extend) တို့ကို သင်ကြားပေးလိုပါသည်။

သင်တန်း ဖွဲ့စည်းပုံ (Class structure)

ကျွန်ုပ်တို့တွင် **ခေါင်းစဉ်အမျိုးမျိုး** (https://missing-semester-my.github.io/2019/) ကို လွှမ်းခြုံထားသည့် သင်ခန်းစာ ၆ ခု ပါဝင်ပါသည်။ သင်ခန်းစာ မှတ်စုများကို အွန်လိုင်းတွင် ရယူနိုင်သော်လည်း မှတ်စုများတွင် ပါဝင်မည် မဟုတ်သည့် လက်တွေ့ ပြသမှု (demo) များစွာကို အတန်းထဲတွင် သင်ကြားပေးသွားမည် ဖြစ်ပါသည်။ သင်ခန်းစာ ရိုက်ကူးချက်များကိုလည်း အွန်လိုင်းတွင် တင်ပေးထားမည် ဖြစ်ပါသည်။

အတန်း တစ်ခုစီကို မိနစ် ၅၀ ကြာ သင်ခန်းစာ ၂ ခုဖြင့် ပိုင်းခြားထားပြီး ကြားထဲတွင် ၁၀ မိနစ် နားချိန် ပါဝင်ပါသည်။ သင်ခန်းစာများသည် အဓိကအားဖြင့် တိုက်ရိုက် လက်တွေ့ ပြသမှုများ ဖြစ်ပြီး ယင်းနောက် လက်တွေ့ လေ့ကျင့်ခန်းများ ပြုလုပ်ရမည် ဖြစ်ပါသည်။ အတန်းတစ်ခုစီ၏ အဆုံးတွင် office-hours ပုံစံဖြင့် လေ့ကျင့်ခန်းများကို စတင် ပြုလုပ်နိုင်ရန် အချိန်တိုတစ်ခု ရရှိနိုင်ပါသည်။

ဤသင်တန်းမှ အကျိုးကျေးဇူး အပြည့်အဝ ရရှိနိုင်ရန် လေ့ကျင့်ခန်း အားလုံးကို မိမိကိုယ်တိုင် ပြုလုပ်ကြည့် သင့်ပါသည်။ ကျွန်ုပ်တို့သည် မိမိ၏ tool များအကြောင်း ပိုမို လေ့လာရန် စိတ်အားထက်သန်မှုများ ပေးစွမ်း မည်ဖြစ်ပြီး၊ မည်သို့ ပြုလုပ်နိုင်သည်ကို ပြသကာ အခြေခံ အချက်အချို့ကို အသေးစိတ် လွှမ်းခြုံ သင်ကြား ပေးမည် ဖြစ်သော်လည်း ရရှိထားသည့် အချိန်အတွင်း အရာအားလုံးကိုတော့ သင်ကြားပေးနိုင်မည် မဟုတ် ပါ။

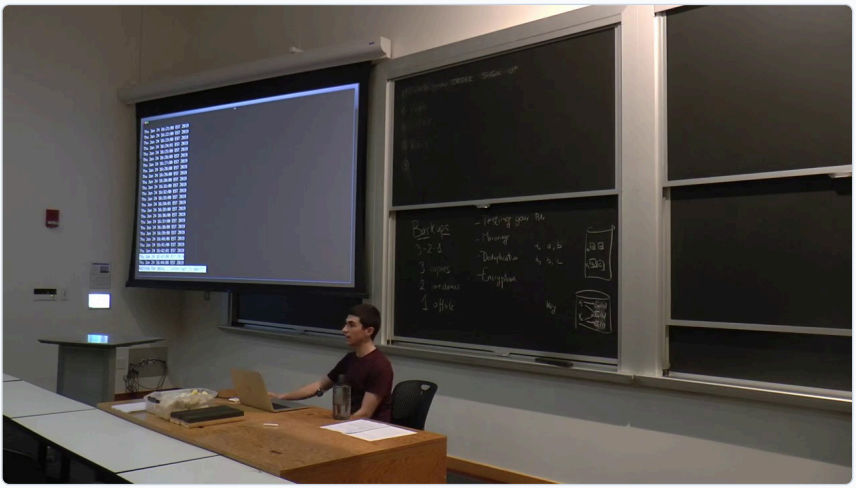
🔗 External Links & References (ပြင်ပ လင့်ခ်များနှင့် ကိုးကားချက်များ)

Resource / Description	URL	Micro QR
1. hacker culture	https://en.wikipedia.org/wiki/Hacker_culture	
2. security hacker	https://en.wikipedia.org/wiki/Security_hacker	
3. ခေါင်းစဉ်အမျိုးမျိုး	https://missing-semester-my.github.io/2019/	

Automation

▶ LECTURE VIDEO REFERENCE

Automation



Scan the QR code below using your mobile camera or click the link to watch this full lecture online:

Direct Online Link:

https://www.youtube.com/watch?v=BaLIAaHz-1k

အချို့သော အခါများတွင် သင်သည် အလုပ်တစ်ခုခု လုပ်ဆောင်ပေးသည့် script တစ်ခုကို ရေးသားပြီး ၎င်းကို ပုံမှန် အချိန်ခြားပြီး (ဥပမာ- backup ပြုလုပ်သည့် လုပ်ဆောင်ချက်ကဲ့သို့) အလိုအလျောက် ပုံမှန် လည်ပတ်စေချင်ပေမည်။ နောက်ကွယ် (background) တွင် ပုံမှန် ပွင့်လာပြီး လည်ပတ်ပေးသည့် သီးသန့် အဆင်ပြေရာ *ad hoc* ဖြေရှင်းနည်းတစ်ခုကို ရေးသား၍လည်း ရနိုင်ပါသည်။ သို့သော်လည်း UNIX စနစ်အများစုတွင် ရိုးရှင်းသော စည်းမျဉ်းများအပေါ် အခြေခံ၍ အနည်းဆုံး တစ်မိနစ်လျှင် တစ်ကြိမ်အထိ လုပ်ဆောင်ချက်များကို ပုံမှန် လည်ပတ်ပေးနိုင်သည့် cron daemon ပါဝင်ပြီးသား ဖြစ်ပါသည်။

UNIX စနစ်အများစုတွင် cron daemon ဖြစ်သည့် `crond` သည် မူလအတိုင်း (by default) လည်ပတ်နေလေ့ရှိသော်လည်း `ps aux | grep crond` ကို အသုံးပြု၍ အမြဲတမ်း စစ်ဆေးကြည့်နိုင်ပါသည်။

crontab

cron အတွက် configuration ဖိုင်ကို `crontab -l` ဖြင့် ကြည့်ရှုနိုင်ပြီး `crontab -e` ဖြင့် ပြင်ဆင်နိုင်သည်။ cron အသုံးပြုသော အချိန်ပုံစံ (time format) တွင် space ဖြင့် ပိုင်းခြားထားသော ဖိန်း (field) ၅ ခု ပါဝင်ပြီး ယင်းနောက်တွင် user နှင့် command တို့ ပါဝင်သည်။

- **minute** - command စတင် လည်ပတ်ရမည့် မိနစ် ဖြစ်ပြီး '0' နှင့် '59' ကြား သတ်မှတ်ရသည်။
- **hour** - command စတင် လည်ပတ်ရမည့် နာရီ ဖြစ်ပြီး ၂၄ နာရီ စနစ်ဖြင့် သတ်မှတ်ရသည်၊ တန်ဖိုးများမှာ 0 မှ 23 ကြား ဖြစ်ရမည် (0 သည် သန်းခေါင်ယံ ဖြစ်သည်)။
- **dom** - ၎င်းသည် Day of Month (လ၏ ရက်စွဲ) ဖြစ်ပြီး command ကို လည်ပတ်စေချင်သည့် ရက် ဖြစ်သည်၊ ဥပမာ- လစဉ် ၁၉ ရက်နေ့တွင် command လည်ပတ်စေချင်ပါက dom သည် 19 ဖြစ်မည်။
- **month** - command လည်ပတ်မည့် လဖြစ်ပြီး ကိန်းဂဏန်း (0-12) ဖြင့်ဖြစ်စေ သို့မဟုတ် လအမည် (ဥပမာ- May) ဖြင့်ဖြစ်စေ သတ်မှတ်နိုင်သည်။
- **dow** - ၎င်းသည် Day of Week (တစ်ပတ်၏ နေ့ရက်) ဖြစ်ပြီး command ကို လည်ပတ်စေချင်သည့် နေ့ ဖြစ်သည်၊ ကိန်းဂဏန်း (0-7) ဖြင့်ဖြစ်စေ သို့မဟုတ် နေ့အမည် (ဥပမာ- sun) ဖြင့်ဖြစ်စေ သတ်မှတ် နိုင်သည်။
- **user** - command ကို လည်ပတ်စေသည့် user ဖြစ်သည်။
- **command** - လည်ပတ်စေချင်သည့် command ဖြစ်သည်။ ဤနေရာတွင် စကားလုံး အများအပြား သို့မဟုတ် space များ ပါဝင်နိုင်သည်။

ကြယ်ပွင့် `*` ကို အသုံးပြုပါက "အားလုံး" ကို ဆိုလိုပြီး ကြယ်ပွင့်နောက်တွင် စလတ်ရှ် နှင့် ကိန်းဂဏန်း ယှဉ်တွဲသုံးပါက n ကြိမ်မြောက် တန်ဖိုးတိုင်းကို ဆိုလိုပါသည်။ ထို့ကြောင့် `*/5` ဆိုသည်မှာ ၅ ကြိမ်တိုင်းတွင် တစ်ကြိမ် ကို ဆိုလိုသည်။ ဥပမာအချို့မှာ-

```
*/5   *    *   *   *       # Every five minutes
  0   *    *   *   *       # Every hour at o'clock
  0   9    *   *   *       # Every day at 9:00 am
  0   9-17 *   *   *       # Every hour between 9:00am and 5:00pm
  0   0    *   *   5       # Every Friday at 12:00 am
  0   0    1   */2 *       # Every other month, the first day, 12:00am
```

အတွေ့ရများသော crontab ဇယားသတ်မှတ်ချက် နမူနာအမြောက်အများကို [crontab.guru](https://crontab.guru/examples.html) တွင် ပိုမို ရှာဖွေကြည့်ရှုနိုင်ပါသည်။

Shell ပတ်ဝန်းကျင် (environment) နှင့် မှတ်တမ်းတင်ခြင်း (logging)

cron ကို အသုံးပြုရာတွင် တွေ့ရလေ့ရှိသော အဓိက အမှားတစ်ခုမှာ ၎င်းသည် အသုံးများသော shell များကဲ့သို့ `.bashrc` ၊ `.zshrc` အစရှိသည့် environment script များကို မူလအတိုင်း ရယူခြင်းမရှိဘဲ output များကိုလည်း မူလအတိုင်း မည်သည့်နေရာတွင်မှ မှတ်တမ်း (log) တင်ထားခြင်း မရှိခြင်းဖြစ်သည်။ အများဆုံး အကြိမ်ရေမှာ တစ်မိနစ်လျှင် တစ်ကြိမ်ဖြစ်ခြင်းနှင့် ပေါင်းစပ်လိုက်သည့်အခါ စတင် အသုံးပြုချိန်တွင် cronscript များကို debug ပြုလုပ်ရန် တော်တော်လေး အခက်တွေ့စေနိုင်ပါသည်။

environment ပြဿနာကို ဖြေရှင်းရန် သင်၏ script များအားလုံးတွင် တိကျသော လမ်းကြောင်း (absolute path) များကို သုံးစွဲရန် သေချာစေပြီး script ကို အောင်မြင်စွာ လည်ပတ်နိုင်ရန် `PATH` ကဲ့သို့သော environment variable များကို ပြင်ဆင်ပါ။ logging ပြုလုပ်ခြင်းကို လွယ်ကူစေရန်အတွက် သင်၏ crontab ကို အောက်ပါ ပုံစံအတိုင်း ရေးသားရန် အကြံပြုပါသည်။

```
* * * * *   user  /path/to/cronscripts/every_minute.sh >> /tmp/cron_every_minut
e.log 2>&1
```

ပြီးလျှင် script ကို သီးခြား ဖိုင်တစ်ခုတွင် ရေးသားပါ။ `>>` သည် ဖိုင်အဆုံးတွင် သွားရောက် ထည့်သွင်း (append) ပေးပြီး `2>&1` သည် `stderr` ကို `stdout` သို့ လမ်းကြောင်းပြောင်းပေးခြင်း (redirect) ဖြစ်ကြောင်း သတိရပါ (ယင်းတို့နှစ်ခုကို သီးခြားစီ ခွဲထားချင်ပါကလည်း ရပါသည်)။

Anacron

cron ကို အသုံးပြုရာတွင် သတိထားရန် အချက်တစ်ခုမှာ cron script လည်ပတ်ရမည့် အချိန်တွင် ကွန်ပျူတာ ပိတ်ထားပါက သို့မဟုတ် asleep ဖြစ်နေပါက script သည် လည်ပတ်မည် မဟုတ်ပေ။ မကြာခဏ လည်ပတ်သည့် လုပ်ဆောင်ချက်များအတွက် ၎င်းသည် အဆင်ပြေနိုင်သော်လည်း ခပ်ကျဲကျဲသာ လည်ပတ်သည့် လုပ်ဆောင်ချက်များအတွက်မူ ယင်းကို မပျက်မကွက် လည်ပတ်စေချင်ပေမည်။ [anacron](https://linux.die.net/man/8/anacron) သည် အကြိမ်ရေကို နေ့ရက်များဖြင့် သတ်မှတ်သည်မှလွဲ၍ `cron` နှင့် သဘောတရား ဆင်တူပါသည်။ cron နှင့်မတူသည်မှာ ၎င်းသည် စက်ကို အစဉ်မပြတ် လည်ပတ်နေသည်ဟု မယူဆထားပါ။ ထို့ကြောင့် ၎င်းကို ၂၄ နာရီပတ်လုံး လည်ပတ်မနေသော စက်များတွင် နေ့စဉ်၊ အပတ်စဉ်၊ နှင့် လစဉ် လုပ်ဆောင်ချက်များကဲ့သို့ ပုံမှန် အလုပ်များကို စီမံခန့်ခွဲရန် အသုံးပြုနိုင်သည်။

လေ့ကျင့်ခန်းများ (Exercises)

1. သင်၏ Downloads ဖိုဒါအတွင်းရှိ ဓာတ်ပုံဖိုင်များ (**MIME types** (https://developer.mozilla.org/en-US/docs/Web/HTTP/Basics_of_HTTP/MIME_types) ကို လေ့လာကြည့်ရှုနိုင်သည် သို့မဟုတ် အသုံးများသော extension များကို ကိုက်ညီစေရန် regular expression ကို အသုံးပြုနိုင်သည်) ကို တစ်မိနစ်လျှင် တစ်ကြိမ် စစ်ဆေးပြီး Pictures ဖိုဒါသို့ ရွှေ့ပြောင်းပေးသည့် script တစ်ခု ပြုလုပ်ပါ။

2. သင်၏ စနစ်အတွင်း ခေတ်မမီတော့သော (outdated) package များကို အပတ်စဉ် စစ်ဆေးပြီး update ပြုလုပ်ရန် အကြောင်းကြားပေးသည့် သို့မဟုတ် အလိုအလျောက် update ပြုလုပ်ပေးသည့် cron script တစ်ခု ရေးသားပါ။

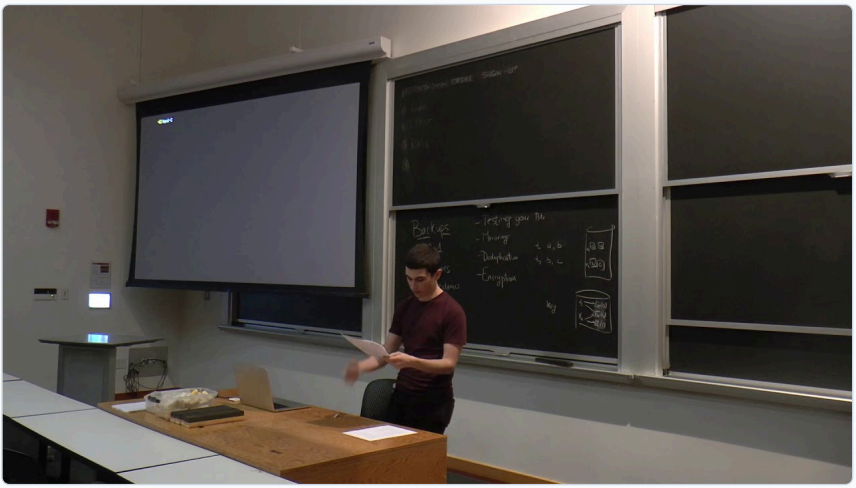
🔗 External Links & References (ပြင်ပ လင့်ခ်များနှင့် ကိုးကားချက်များ)

Resource / Description	URL	Micro QR
1. crontab.guru	https://crontab.guru/examples.html	
2. anacron	https://linux.die.net/man/8/anacron	
3. MIME types	https://developer.mozilla.org/en-US/docs/Web/HTTP/Basics_of_HTTP/MIME_types	

Backups

▶ LECTURE VIDEO REFERENCE

Backups



Scan the QR code below using your mobile camera or click the link to watch this full lecture online:

Direct Online Link:

https://www.youtube.com/watch?v=IrpqYF8tcYQ

လူ အမျိုးအစား နှစ်မျိုး ရှိပါသည်-

- ဒေတာ အရန်သိမ်းဆည်းမှု (Backup) ပြုလုပ်သူများ
- ဒေတာ အရန်သိမ်းဆည်းမှု ပြုလုပ်လာမည့်သူများ

သင် ပိုင်ဆိုင်ထားပြီး အရန်သိမ်းဆည်း မထားသည့် မည်သည့် ဒေတာမဆို မည်သည့် အချိန်တွင်မဆို ထာဝရ ပျောက်ကွယ်သွားနိုင်သည့် ဒေတာ ဖြစ်ပါသည်။ ဤသင်ခန်းစာတွင် အသုံးဝင်သော ဒေတာ အရန်သိမ်းဆည်း

ခြင်းဆိုင်ရာ အခြေခံ အချက်များနှင့် ချဉ်းကပ်ပုံ အချို့၏ အားနည်းချက်/အမှားအယွင်းများကို ဆွေးနွေးဖော်ပြသွားမည် ဖြစ်ပါသည်။

3-2-1 နည်းဥပဒေ (3-2-1 Rule)

3-2-1 နည်းဥပဒေ (https://www.us-cert.gov/sites/default/files/publications/data_backup_options.pdf) ဆိုသည်မှာ မိမိ၏ ဒေတာများကို အရန်သိမ်းဆည်းရန် ယေဘုယျအားဖြင့် အကြံပြုထားသော မဟာဗျူဟာ တစ်ခု ဖြစ်ပါသည်။ ယင်းတွင် သင့်ထံ၌ အောက်ပါအတိုင်း ရှိသင့်သည်ဟု ဖော်ပြထားပါသည်-

- မိမိ ဒေတာ၏ **မိတ္တူ အနည်းဆုံး ၃ စုံ (3 copies)**
- မတူညီသော **သိုလှောင်မှု မီဒီယာ (different mediums)** တွင် သိမ်းဆည်းထားသော မိတ္တူ **၂ စုံ**
- မိတ္တူများအနက် **၁ စုံ** ကို **မူလ နေရာမဟုတ်သည့် တခြားနေရာ (offsite)** တွင် သိမ်းဆည်းထားခြင်း

ဤအကြံပြုချက်၏ အဓိက ရည်ရွယ်ချက်မှာ ဥများ အားလုံးကို ခြင်းတောင်း တစ်ခုတည်းထဲ မထည့်ထားရန် ဖြစ်ပါသည်။ မတူညီသော စက်ပစ္စည်း/ဒစ်ခ် (devices/disks) ၂ ခုတွင် သိမ်းဆည်းထားခြင်းဖြင့် ဟာ့ဒ်ဝဲ တစ်ခု ပျက်စီးရုံဖြင့် သင်၏ ဒေတာအားလုံး ဆုံးရှုံးမသွားစေရန် အာမခံနိုင်ပါသည်။ ထို့အတူ မိမိ၏ တစ်ခုတည်းသော အရန်သိမ်းဆည်းထားသည့် မိတ္တူကို အိမ်တွင်သာ သိမ်းဆည်းထားပြီး အိမ် မီးလောင်သွားလျှင် သို့မဟုတ် သူခိုး ခိုးခံရလျှင် သင် အရာအားလုံး ဆုံးရှုံးရမည် ဖြစ်ပါသည်။ အခြားနေရာတွင် သိမ်းဆည်းထားသည့် Offsite မိတ္တူမှာ ယင်းကဲ့သို့ အခြေအနေများအတွက် ရည်ရွယ်ထားခြင်း ဖြစ်ပါသည်။ မူလနေရာတွင် သိမ်းဆည်းသည့် (Onsite) အရန်မိတ္တူများသည် သုံးစွဲရလွယ်ကူမှုနှင့် မြန်ဆန်မှုကို ပေးစွမ်းပြီး မူလနေရာမဟုတ်သည့် (Offsite) မိတ္တူများသည် ဘေးအန္တရာယ် ကြုံတွေ့ရပါက ပြန်လည် ရယူနိုင်သည့် ကြံ့ကြံ့ခံနိုင်စွမ်း (resiliency) ကို ပေးစွမ်းနိုင်ပါသည်။

မိမိ၏ အရန်မိတ္တူများကို စမ်းသပ်စစ်ဆေးခြင်း (Testing your backups)

ဒေတာ အရန်သိမ်းဆည်းရာတွင် တွေ့ရလေ့ရှိသည့် အမှားတစ်ခုမှာ စနစ်က လုပ်ဆောင်နေသည်ဟု ဆိုသမျှကို မျက်စိမှိတ် ယုံကြည်ပြီး ဒေတာများကို စနစ်တကျ ပြန်လည်ရယူနိုင်ခြင်း ရှိမရှိ (verify) မစစ်ဆေးခြင်း ဖြစ်ပါသည်။ Toy Story 2 ရုပ်ရှင်သည် တစ်ခုလုံးနီးပါး ဆုံးရှုံးခဲ့ရ ဖြစ်ခဲ့ဖူးပြီး သူတို့၏ အရန်သိမ်းဆည်းမှုများ မလုပ်ဆောင်ခဲ့ဘဲ နောက်ဆုံးတွင် **ကံကောင်းမှု** (https://www.youtube.com/watch?v=8dhp_20j0Ys) ကြောင့်သာ ကယ်တင်နိုင်ခဲ့ခြင်း ဖြစ်ပါသည်။

ဗားရှင်း ခွဲခြားသိမ်းဆည်းခြင်း (Versioning)

RAID (https://en.wikipedia.org/wiki/RAID) သည် ဒေတာ အရန်သိမ်းဆည်းမှု မဟုတ်ကြောင်းနှင့် ယေဘုယျအားဖြင့် **mirroring သည် အရန်သိမ်းဆည်းမှု နည်းလမ်းတစ်ခု မဟုတ်ကြောင်း** နားလည်ထားသင့်ပါသည်။ မိမိ၏ ဖိုင်များကို အခြားနေရာ တစ်ခုခုသို့ Sync လုပ်ရုံမျှဖြင့် အောက်ပါ အခြေအနေမျိုးစုံတွင် ကူညီပေးနိုင်မည် မဟုတ်ပါ-

- ဒေတာ ပျက်စီးယိုယွင်းခြင်း (Data corruption)
- မလိုလားအပ်သော အန္တရာယ်ရှိ ဆော့ဖ်ဝဲများ (Malicious software)
- ဖိုင်များကို မှားယွင်း ဖျက်ဆီးမိခြင်း (Deleting files by mistake)

မိမိဒေတာ၏ ပြောင်းလဲမှုများသည် အရန်သိမ်းဆည်းထားသည့် နေရာသို့ပါ ကူးစက် ပျံ့နှံ့သွားပါက ဤအခြေအနေများတွင် ပြန်လည် ရယူနိုင်တော့မည် မဟုတ်ပါ။ Dropbox, Google Drive, OneDrive စသည့် Cloud Storage ဝန်ဆောင်မှု အများအပြားတွင် ဤကဲ့သို့ ဖြစ်လေ့ရှိသည်ကို သတိပြုပါ။ ယင်းတို့အနက် အချို့သည် ဖျက်လိုက်သော ဒေတာများကို အချိန်တိုအတွင်း ထိန်းသိမ်းပေးထားသော်လည်း ဖိုင်ပမာဏ အများအပြားကို ပြန်လည်ရယူရန်အတွက် ယင်း interface များကို အသုံးပြုရခြင်းမှာ အဆင်ပြေလှသည် မဟုတ်ပါ။

ဤသို့ ပျက်စီးဆုံးရှုံးမှုမျိုး မဖြစ်စေရန် စနစ်ကျသော အရန်သိမ်းဆည်းမှု စနစ်တစ်ခုတွင် ဗားရှင်း ခွဲခြားခြင်း (versioned) ပါဝင်သင့်ပါသည်။ အချိန်ကာလ အလိုက် ကွဲပြားသော snapshot များကို ပေးဆောင်ထားခြင်းဖြင့် ဆုံးရှုံးသွားသော မည်သည့် ဒေတာကိုမဆို ပြန်လည် ရယူရန် လွယ်ကူစွာ ရှာဖွေနိုင်မည် ဖြစ်ပါသည်။ ဤကဲ့သို့သော အမျိုးအစားတွင် လူသိအများဆုံး ဆော့ဖ်ဝဲမှာ macOS ၏ Time Machine ဖြစ်ပါသည်။

ထပ်နေသော ဒေတာများကို ဖယ်ရှားခြင်း (Deduplication)

မိမိ၏ ဒေတာများကို မိတ္တူ အများအပြား ကူးယူထားခြင်းသည် ဒစ်ခ် ပမာဏ (disk space) အလွန် ကုန်ကျနိုင်ပါသည်။ သို့သော်လည်း ဗားရှင်းတစ်ခုမှ နောက်တစ်ခုသို့ ပြောင်းလဲရာတွင် ဒေတာ အများစုမှာ အတူတူပင် ဖြစ်ပြီး ထပ်မံ ပေးပို့ရန် မလိုပါ။ ဤနေရာတွင် **data deduplication** (https://en.wikipedia.org/wiki/Data_deduplication) (ထပ်နေသော ဒေတာများကို ဖယ်ရှားခြင်း) နည်းပညာ ဝင်ရောက်လာပြီး၊ သိမ်းဆည်းပြီးသား ဒေတာများကို မှတ်တမ်းတင်ထားခြင်းဖြင့် ဗားရှင်းတစ်ခုမှ နောက်တစ်ခုသို့ ပြောင်းလဲသွားသော အချက်အလက်များကိုသာ သိမ်းဆည်းသည့် **incremental backups** များကို ပြုလုပ်နိုင်ပါသည်။ ယင်းက ပထမဆုံး မိတ္တူ လွန်ပြီးနောက် အရန်သိမ်းဆည်းရန် လိုအပ်သော နေရာပမာဏကို သိသိသာသာ လျော့ချပေးပါသည်။

ဒေတာ ဝှက်စာစနစ် (Encryption)

Cloud Provider များကဲ့သို့ မယုံကြည်ရသော အပြင်ပန်း ဝန်ဆောင်မှုများသို့ အရန်သိမ်းဆည်းနေပါက မိမိဒေတာကို *မူရင်းအတိုင်း (as is)* ကူးယူ သိမ်းဆည်းထားလျှင် မလိုလားအပ်သူများ ကြည့်ရှုသွားနိုင်ခြေ ရှိသည်ကို ထည့်သွင်း စဉ်းစားသင့်ပါသည်။ အခွန်ဆိုင်ရာ Document များကဲ့သို့သော Document များသည် အကဲဆတ်သော အချက်အလက်များဖြစ်ပြီး သာမန် စာသား (plain format) အဖြစ် အရန်မသိမ်းဆည်းသင့်ပါ။ ဤသည်ကို ကာကွယ်ရန် အရန်သိမ်းဆည်းမှု နည်းလမ်း အများအပြားသည် ဒေတာများကို Server သို့ မပေးပို့မီ Encryption ပြုလုပ်ပေးသည့် **client side encryption** ကို ထောက်ပံ့ပေးထားကြပါသည်။ ထိုနည်းဖြင့် Server သည် သိမ်းဆည်းထားသော ဒေတာများကို ဖတ်ရှုနိုင်မည် မဟုတ်ဘဲ သင့်ထံတွင် ရှိသည့် လျှို့ဝှက်ကီး (secret key) ဖြင့်သာ Decrypt ပြုလုပ် ဖတ်ရှုနိုင်မည် ဖြစ်ပါသည်။

ဖြည့်စွက် အချက်အနေဖြင့် သင့်ဒစ်ခ် (သို့မဟုတ် home partition) ကို Encrypt မလုပ်ထားပါက သင့်ကွန်ပျူတာကို ရရှိသွားသူ မည်သူမဆို သုံးစွဲသူ မူပိုင်ခွင့် ဝင်ရောက်မှု ထိန်းချုပ်ချက်များကို ကျော်လွှန်၍ သင့်ဒေတာများကို ဖတ်ရှုနိုင်မည် ဖြစ်ပါသည်။ ခေတ်မီ ဟာ့ဒ်ဝဲများသည် Encrypted ဒေတာများကို မြန်ဆန် ထိရောက်စွာ ဖတ်ရှု ရေးသားနိုင်သောကြောင့် **full disk encryption** ကို ဖွင့်လှစ် အသုံးပြုရန် စဉ်းစားသင့်ပါသည်။

ဖြည့်စွက်ရေးသားခြင်း သာပြုနိုင်ခြင်း (Append only)

ယခုအချိန်ထိ သုံးသပ်ခဲ့သော အချက်များသည် ဟာ့ဒ်ဝဲ ပျက်စီးမှု သို့မဟုတ် သုံးစွဲသူ၏ အမှားများကိုသာ အဓိကထားပြီး မလိုလားအပ်သော အန္တရာယ်ရှိသူတစ်ဦးက သင့်ဒေတာများကို ဖျက်ဆီးလိုပါက မည်သို့ဖြစ်မည်ကို ဖြေရှင်းနိုင်ခြင်း မရှိသေးပါ။ ဆိုလိုသည်မှာ တစ်စုံတစ်ယောက်က သင့်စနစ်ကို Hack လုပ်လိုက်ပါက သင်ဂရုစိုက်ရသော ဒေတာ မိတ္တူ အားလုံးကို ဖျက်ဆီးပစ်နိုင်မည်လား။ အကယ်၍ ထိုအခြေအနေအတွက် စိုးရိမ်ပါက Append-only အရန်သိမ်းဆည်းမှု နည်းလမ်းတစ်မျိုး လိုအပ်မည် ဖြစ်ပါသည်။ ယေဘုယျအားဖြင့် ဒေတာအသစ်များ ပေးပို့ခြင်းကို ခွင့်ပြုသော်လည်း ရှိပြီးသား ဒေတာများကို ဖျက်ဆီးခြင်းကို ငြင်းပယ်သည့် Server တစ်ခု ရှိနေခြင်းကို ဆိုလိုပါသည်။ ပုံမှန်အားဖြင့် သုံးစွဲသူများတွင် Key နှစ်ခု ရှိကြပြီး၊ အရန်မိတ္တူ အသစ်များ ဖန်တီးခြင်းကို ထောက်ပံ့သည့် append-only key နှင့် မလိုအပ်တော့သည့် အဟောင်းများကို ဖျက်ဆီးခွင့်ပြုသည့် full access key တို့ ဖြစ်ကြပါသည်။ နောက်ဆုံး key ကိုတော့ offline တွင် သိမ်းဆည်းထားလေ့ ရှိပါသည်။

အန္တရာယ်ရှိ သုံးစွဲသူက သင့်ဒေတာကို ဖျက်ဆီးခြင်းမှ တားဆီးထားစဉ်မှာပင် ပြောင်းလဲမှုများ ပြုလုပ်နိုင်ရန် လိုအပ်သည့်အတွက် ဤသည်မှာ တော်တော်လေး ခက်ခဲသော အခြေအနေတစ်ခု ဖြစ်သည်ကို သတိပြုပါ။ လက်ရှိတွင် စီးပွားရေးအရ အသုံးပြုနိုင်သော နည်းလမ်းများတွင် **Tarsnap** (https://www.tarsnap.com/) နှင့် **Borgbase** (https://www.borgbase.com/) တို့ ပါဝင်ပါသည်။

ထပ်မံ စဉ်းစားသင့်သည့် အချက်များ (Additional considerations)

ထပ်မံ လေ့လာ စဉ်းစားသင့်သည့် အခြား အချက်အချို့မှာ-

- **ပုံမှန် အချိန်အပိုင်းအခြားအလိုက် အရန်သိမ်းခြင်း (Periodic backups)**: ခေတ်နောက်ကျနေသော အရန်မိတ္တူများသည် အသုံးမဝင် သလောက် ဖြစ်သွားနိုင်ပါသည်။ ပုံမှန် အချိန်မှန် အရန်သိမ်းဆည်းခြင်းကို မိမိ စနစ်အတွက် ထည့်သွင်း စဉ်းစားသင့်ပါသည်။
- **Boot တိုက်ရိုက် လုပ်နိုင်သော အရန်မိတ္တူများ (Bootable backups)**: ပရိုဂရမ် အချို့သည် မိမိ၏ ဒစ်ခ် တစ်ခုလုံးကို Clone ကူးယူခွင့် ပြုထားပါသည်။ ထို့ကြောင့် စနစ်တစ်ခုလုံး၏ မိတ္တူပါဝင်သော Image တစ် ခုရရှိပြီး ယင်းမှ တိုက်ရိုက် Boot တက်နိုင်မည် ဖြစ်ပါသည်။
- **ကွဲပြားသော အရန်သိမ်းဆည်းမှု မဟာဗျူဟာများ (Differential backup strategies)**: မိမိ၏ ဒေတာ အားလုံးကို အရေးပါမှု အတူတူ ထားရှိချင်မှ ထားရှိပါမည်။ ဒေတာ အမျိုးအစား မတူညီမှုအပေါ် မူတည်၍ ကွဲပြားသော အရန်သိမ်းဆည်းမှု မူဝါဒများကို သတ်မှတ်နိုင်ပါသည်။
- **Append only အရန်သိမ်းဆည်းမှုများ (Append only backups)**: ဖြည့်စွက် စဉ်းစားရမည့် အချက်မှာ သင့်စက်ကို အန္တရာယ်ရှိသူများ ရရှိသွားပါက အရန်သိမ်းထားသည်များကို မဖျက်နိုင်စေရန် အရန် Repository များတွင် Append only စနစ်ကို သတ်မှတ်ထားခြင်း ဖြစ်ပါသည်။

ဝဘ်ဝန်ဆောင်မှုများ (Webservices)

သင် အသုံးပြုသည့် ဒေတာ အားလုံးသည် ဟာ့ဒစ်ခ်ပေါ်တွင်သာ ရှိနေသည် မဟုတ်ပါ။ အကယ်၍ သင်သည် **ဝ ဘ်ဝန်ဆောင်မှုများ (webservices)** ကို အသုံးပြုပါက Google Docs ပို့ချချက်များ သို့မဟုတ် Spotify ဖွင့်ရန် သီချင်းစာရင်းများကဲ့သို့သော သင်ဂရုစိုက်ရသည့် ဒေတာအချို့သည် အွန်လိုင်းတွင် သိမ်းဆည်းထားခြင်း ဖြစ် နိုင်ပါသည်။ မေ့လျော့ရလွယ်သည့် အခြား သာဓကတစ်ခုမှာ Gmail ကဲ့သို့သော ဝဘ်မှတစ်ဆင့် အသုံးပြုနိုင် သည့် အီးမေးလ် အကောင့်များ ဖြစ်ပါသည်။ ဤအခြေအနေများတွင် အရန်သိမ်းဆည်းမှု နည်းလမ်း ရှာဖွေ ခြင်းမှာ အနည်းငယ် ပိုမို ရှုပ်ထွေးနိုင်ပါသည်။ သို့သော်လည်း တိုက်ရိုက်ဖြစ်စေ၊ API မှတစ်ဆင့်ဖြစ်စေ မိမိ၏ ဒေတာများကို ဒေါင်းလုဒ်ဆွဲနိုင်စေမည့် ဝဘ်ဝန်ဆောင်မှုများစွာ ရှိပါသည်။ Gmail အတွက် **gmvault** (https://github.com/gaubert/gmvault) ကဲ့သို့သော Tool များကို အီးမေးလ် ဖိုင်များကို သင့်ကွန်ပျူတာသို့ ဒေါင်းလုဒ် ဆွဲရန် အသုံးပြုနိုင်ပါသည်။

ဝဘ်စာမျက်နှာများ (Webpages)

ထို့အတူ အရည်အသွေးမြင့် အကြောင်းအရာ အချို့ကို အွန်လိုင်းတွင် ဝဘ်စာမျက်နှာများ အဖြစ် တွေ့ရှိနိုင် ပါသည်။ ထိုအကြောင်းအရာသည် Static ဖြစ်ပါက ဝဘ်ဆိုက်နှင့် ယင်း၏ Attachment အားလုံးကို Save လုပ် ရုံမျှဖြင့် လွယ်ကူစွာ အရန်သိမ်းဆည်းနိုင်ပါသည်။ အခြား နည်းလမ်းတစ်ခုမှာ မီဒီယာ ပုံစံမျိုးစုံကို ထိန်းသိမ်း

စောင့်ရှောက်ခြင်းအပေါ် အဓိကထားသည့် အကျိုးအမြတ်မယူသော အဖွဲ့အစည်းတစ်ခုဖြစ်သော **Internet Archive** (https://archive.org/) မှ စီမံခန့်ခွဲသည့် World Wide Web ၏ ကြီးမားလှသော ဒစ်ဂျစ်တယ် မော်ကွန်းတိုက်ကြီး ဖြစ်သည့် **Wayback Machine** (https://archive.org/web/) ဖြစ်ပါသည်။ Wayback Machine သည် ဝဘ်စာမျက်နှာများကို ဖမ်းယူ (capture) ပြီး မော်ကွန်းတင် ထိန်းသိမ်းနိုင်စေကာ နောက်ပိုင်းတွင် ထိုဝဘ်ဆိုက်အတွက် သိမ်းဆည်းထားသော snapshot များအားလုံးကို ပြန်လည် ထုတ်ယူနိုင်စေပါသည်။ အကယ်၍ ယင်းကို အသုံးဝင်သည်ဟု ယူဆပါက စီမံကိန်းသို့ လှူဒါန်းရန် (https://archive.org/donate/) စဉ်းစားပါ။

အရင်းအမြစ်များ (Resources)

ကျွန်ုပ်တို့ ကိုယ်တိုင် အသုံးပြုခဲ့ပြီး စိတ်ချလက်ချ အကြံပြုနိုင်သော အရန်သိမ်းဆည်းမှု ပရိုဂရမ်များနှင့် ဝန်ဆောင်မှု အချို့မှာ-

- **Tarsnap** (https://www.tarsnap.com/) - လုံခြုံရေးအတွက် အလွန် သတိကြီးသူများအတွက် ထပ်နေသည်များ ဖယ်ရှားထားပြီး Encrypt လုပ်ထားသော အွန်လိုင်း အရန်သိမ်းဆည်းမှု ဝန်ဆောင်မှု။
- **Borg Backup** (https://borgbackup.readthedocs.io) - ချုံ့ခြင်း (compression) နှင့် စိစစ်ထားသော Encryption ကို ထောက်ပံ့ပေးသည့် ထပ်နေသည်များ ဖယ်ရှားထားသော အရန်သိမ်းဆည်းမှု ပရိုဂရမ်။ Cloud Provider လိုအပ်ပါက **BorgBase** (https://www.borgbase.com/) သည် လူကြိုက်များသော ရွေးချယ်မှု တစ်ခု ဖြစ်ပါသည်။
- **rsync** (https://rsync.samba.org/) သည် မြန်ဆန်သော incremental ဖိုင် ကူးပြောင်းမှုကို ထောက်ပံ့ပေးသည့် Utility တစ်ခု ဖြစ်ပါသည်။ ယင်းသည် အပြည့်အဝ အရန်သိမ်းဆည်းမှု နည်းလမ်း မဟုတ်ပါ။
- **rclone** (https://rclone.org/) သည် rsync နှင့် ဆင်တူသော်လည်း Amazon S3, Dropbox, Google Drive, rsync.net စသည့် Cloud Storage Provider များအတွက် ဖြစ်ပါသည်။ Remote folder များကို Client-side Encryption ပြုလုပ်ခြင်းကို ထောက်ပံ့ပေးပါသည်။

လေ့ကျင့်ခန်းများ (Exercises)

1. မိမိ၏ ဒေတာများကို မည်သို့ အရန်သိမ်းဆည်းနေသလဲ (သို့မဟုတ် မသိမ်းဆည်းဘဲ ထားသလဲ) ဆိုသည်ကို စဉ်းစားကြည့်ပြီး ယင်းကို ပြင်ဆင်/တိုးမြှင့်ရန် ဆောင်ရွက်ပါ။
2. မိမိ၏ အီးမေးလ် အကောင့်များကို မည်သို့ အရန်သိမ်းဆည်းရမည်ကို ရှာဖွေပါ။
3. မိမိ မကြာခဏ အသုံးပြုလေ့ရှိသည့် ဝဘ်ဝန်ဆောင်မှု တစ်ခု (Spotify, Google Music စသည်) ကို ရွေးချယ်ပြီး မိမိ ဒေတာများကို အရန်သိမ်းဆည်းနိုင်မည့် နည်းလမ်းများကို ရှာဖွေပါ။ ရရှိနိုင်သော API များပေါ်တွင် အခြေခံ၍ လူအများက ပြုလုပ်ထားပြီးဖြစ်သော Tool များ (**youtube-dl** (https://ytdl-org.github.io/youtube-dl/) ကဲ့သို့သော) ဖြေရှင်းချက်များ ရှိလေ့ ရှိပါသည်။

4. နှစ်များစွာအတွင်း မိမိ ခေါက်တုံခေါက်ပြန် ဝင်ရောက်ကြည့်ရှုခဲ့သည့် ဝဘ်ဆိုက်တစ်ခုကို စဉ်းစားပြီး ယင်းကို **archive.org** (https://archive.org/web/) တွင် ရှာဖွေကြည့်ပါ။ ယင်းတွင် ဗားရှင်း ပမာဏ မည်မျှ ရှိသနည်း။

5. Deduplication ကို ထိရောက်စွာ အကောင်အထည်ဖော်နိုင်သော နည်းလမ်းတစ်ခုမှာ Hardlink များကို အသုံးပြုခြင်း ဖြစ်ပါသည်။ Symbolic link (Soft link သို့မဟုတ် Symlink ဟုလည်း ခေါ်သည်) ဆိုသည်မှာ အခြား ဖိုင် သို့မဟုတ် ဖိုဒါကို ညွှန်းဆိုထားသည့် ဖိုင်တစ်ခု ဖြစ်သော်လည်း၊ Hardlink သည် Pointer ၏ ပုံတူ မိတ္တူတစ်စုံ ဖြစ်ပါသည် (ယင်းသည် မူလ inode ကိုပင် အသုံးပြုပြီး ဒစ်ခ်ပေါ်ရှိ တူညီသော နေရာကို ညွှန်းဆိုပါသည်)။ ထို့ကြောင့် မူရင်းဖိုင်ကို ဖျက်လိုက်ပါက Symlink သည် အလုပ်လုပ်တော့မည် မဟုတ်သော်လည်း Hardlink ကမူ ဆက်လက် အလုပ်လုပ်နေမည် ဖြစ်ပါသည်။ သို့သော် Hardlink များသည် ဖိုင် များအတွက်သာ အလုပ်လုပ်ပါသည်။ Hardlink များ ဖန်တီးရန် `ln` command ကို အသုံးပြုကြည့်ပါ၊ ယင်းတို့ကို `ln -s` ဖြင့် ဖန်တီးထားသော Symlink များနှင့် နှိုင်းယှဉ်ကြည့်ပါ။ (macOS တွင် gnu coreutils သို့မဟုတ် hln package ကို ထည့်သွင်းရန် လိုအပ်မည် ဖြစ်ပါသည်)။

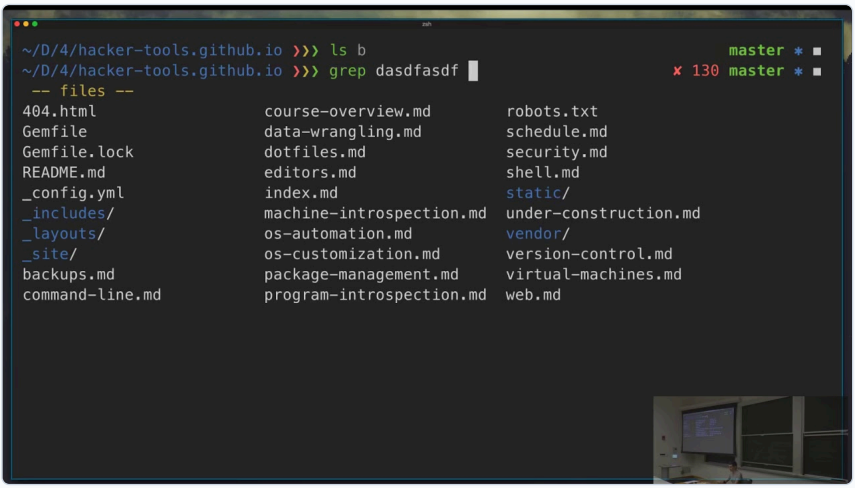
🔗 External Links & References (ပြင်ပ လင့်ခ်များနှင့် ကိုးကားချက်များ)

Resource / Description	URL	Micro QR
1. 3-2-1 နည်းဥပဒေ	https://www.us-cert.gov/sites/default/files/publications/data_backup_options.pdf	
2. ကံကောင်းမှု	https://www.youtube.com/watch?v=8dhp_20j0Ys	
3. RAID	https://en.wikipedia.org/wiki/RAID	
4. data deduplication	https://en.wikipedia.org/wiki/Data_deduplication	
5. Tarsnap	https://www.tarsnap.com/	
6. Borgbase	https://www.borgbase.com/	
7. gmvault	https://github.com/gaubert/gmvault	
8. Internet Archive	https://archive.org/	
9. Wayback Machine	https://archive.org/web/	
10. လှူဒါန်းရန်	https://archive.org/donate/	
11. Borg Backup	https://borgbackup.readthedocs.io	
12. rsync	https://rsync.samba.org/	
13. rclone	https://rclone.org/	
14. youtube-dl	https://ytdl-org.github.io/youtube-dl/	

Command-line ပတ်ဝန်းကျင်

▶ LECTURE VIDEO REFERENCE

Command-line ပတ်ဝန်းကျင်



Scan the QR code below using your mobile camera or click the link to watch this full lecture online:

Direct Online Link:

https://www.youtube.com/watch?v=i0rf1gpKL1E

Aliases နှင့် Functions များ

မိမိစိတ်ကူးကြည့်နိုင်သည့်အတိုင်း flag အများအပြား သို့မဟုတ် ရှည်လျားသော option များပါဝင်သည့် command ရှည်ကြီးများကို ခဏခဏ ရိုက်ထည့်နေရခြင်းသည် မောပန်းစရာကောင်းလှပါသည်။ သို့သော်လည်း shell အများစုသည် **aliasing** (အမည်ပြောင် သတ်မှတ်ခြင်း) ကို ထောက်ပံ့ပေးထားကြ

ပါသည်။ ဥပမာအားဖြင့် bash တွင် alias တစ်ခု၏ တည်ဆောက်ပုံမှာ အောက်ပါအတိုင်း ဖြစ်သည် (`=` သင်္ကေတ၏ ဘေးတစ်ဖက်တစ်ချက်တွင် space ခြားထားခြင်း မရှိသည်ကို သတိပြုပါ) -

```
alias alias_name="command_to_alias"
```

Alias များတွင် အဆင်ပြေစေသော အင်္ဂါရုပ်များစွာ ရှိပါသည် -

```
# Alias can summarize good default flags
alias ll="ls -lh"

# Save a lot of typing for common commands
alias gc="git commit"

# Alias can overwrite existing commands
alias mv="mv -i"
alias mkdir="mkdir -p"

# Alias can be composed
alias la="ls -A"
alias lla="la -l"

# To ignore an alias run it prepended with \
\ls
# Or can be disabled using unalias
unalias la
```

သို့သော်လည်း အခြေအနေများစွာတွင် alias များသည် ကန့်သတ်ချက် ရှိနိုင်ပါသည်၊ အထူးသဖြင့် argument များကို လက်ခံရယူသော command များကို ဆက်တိုက် တွဲရေးရန် ကြိုးစားသည့်အခါမျိုးတွင် ဖြစ်ပါသည်။ ယင်းအစား အခြားရွေးချယ်စရာတစ်ခုမှာ alias များနှင့် စိတ်ကြိုက် shell script များကြား လမ်းဝက်နေရာတွင် ရှိသော **functions** များ ဖြစ်ကြပါသည်။

Directory တစ်ခု တည်ဆောက်ပြီး ယင်းအတွင်းသို့ ချက်ချင်း ဝင်ရောက်သွားသည့် ဥပမာ function တစ်ခုမှာ အောက်ပါအတိုင်း ဖြစ်ပါသည် -

```
mcd () {
    mkdir -p $1
    cd $1
}
```

Alias များနှင့် function များသည် default အားဖြင့် shell session များ ကုန်ဆုံးသွားပါက ပျောက်ပျက်သွားမည် ဖြစ်သည်။ Alias တစ်ခုကို အမြဲတမ်း တည်ရှိနေစေရန် (persistent ဖြစ်စေရန်) ယင်းကို `.bashrc` သို့မဟုတ် `.zshrc` ကဲ့သို့သော shell startup script ဖိုင်များထဲတွင် ထည့်သွင်းပေးရန် လိုအပ်ပါသည်။ ကျွန်ုပ်၏ အကြံပြုချက်မှာ ၎င်းတို့ကို `.alias` ဖိုင်တစ်ခုထဲတွင် သီးသန့် ရေးသားပြီး အခြား shell config ဖိုင်များမှ ထိုဖိုင်ကို `source` လုပ်၍ ခေါ်ယူသုံးစွဲရန် ဖြစ်ပါသည်။

Shell များ နှင့် Framework များ

Shell နှင့် scripting သင်ခန်းစာတွင် ကျွန်ုပ်တို့သည် `bash` shell အကြောင်းကို လေ့လာခဲ့ကြပါသည်၊ အကြောင်းမှာ ယင်းသည် နေရာတိုင်းလိုလိုတွင် အများဆုံး သုံးကြပြီး စနစ်အများစုတွင်လည်း default ရွေးချယ်မှုအဖြစ် ပါဝင်သောကြောင့် ဖြစ်ပါသည်။ သို့သော်လည်း ယင်းသည် တစ်ခုတည်းသော ရွေးချယ်မှု တော့ မဟုတ်ပါ။

ဥပမာအားဖြင့် `zsh` shell သည် `bash` ၏ superset တစ်ခုဖြစ်ပြီး အသင့်သုံးနိုင်သော အဆင်ပြေသည့် အင်္ဂါရုပ်များစွာကို ထောက်ပံ့ပေးထားပါသည် -

- ပိုမိုကောင်းမွန်သော globbing, `**`
- Inline globbing/wildcard စာလုံးချဲ့ထွင်မှု
- စာလုံးပေါင်း မှားယွင်းမှု ပြင်ဆင်ပေးခြင်း
- ပိုမိုကောင်းမွန်သော tab ဖြင့် အလိုအလျောက် ဖြည့်စွက်ခြင်း/ရွေးချယ်ခြင်း (tab completion/selection)
- Path ချဲ့ထွင်မှု (`cd /u/lo/b` သည် `/usr/local/bin` အဖြစ် ကျယ်ပြန့်သွားမည်)

ထို့အပြင် shell အများအပြားကို **framework များ** ဖြင့်လည်း ပိုမိုကောင်းမွန်အောင် ပြုလုပ်နိုင်ပါသည်။ **prezto** (https://github.com/sorin-ionescu/prezto) သို့မဟုတ် **oh-my-zsh** (https://github.com/robbyrussell/oh-my-zsh) ကဲ့သို့သော ရေပန်းစားသည့် အထွေထွေ framework များ ရှိသကဲ့သို့၊ **zsh-syntax-highlighting** (https://github.com/zsh-users/zsh-syntax-highlighting) သို့မဟုတ် **zsh-history-substring-search** (https://github.com/zsh-users/zsh-history-substring-search) ကဲ့သို့ သီးခြား feature များကို အဓိကထားသည့် ပိုမိုသေးငယ်သော framework များလည်း ရှိပါသည်။ **fish** (https://fishshell.com/) ကဲ့သို့သော အခြား shell များတွင်မူ ဤအသုံးပြုရလွယ်ကူသော feature အများအပြားကို default အနေဖြင့် ထည့်သွင်းပေးထားပါသည်။ ၎င်းတို့တွင် ပါဝင်သော feature အချို့မှာ အောက်ပါအတိုင်း ဖြစ်သည် -

- ညာဘက် prompt
- Command စာကြောင်း ကာလာရောင်စုံပြသမှု (syntax highlighting)
- History ဖြတ်ပိုင်းစာလုံးများ ရှာဖွေမှု (history substring search)

- manpage အပေါ် အခြေခံသည့် flag အလိုအလျောက် ဖြည့်စွက်မှုများ
- ပိုမိုဉာဏ်ကောင်းသော autocompletion
- Prompt သင်းမ် (themes) များ

ဤ framework များကို အသုံးပြုရာတွင် သတိပြုရမည့် အချက်တစ်ခုမှာ ၎င်းတို့ runသည့် code များသည် သေချာစွာ စနစ်တကျ ပြင်ဆင်ထားခြင်း (optimize လုပ်ထားခြင်း) မရှိပါက သို့မဟုတ် စာကြောင်းများ အလွန်များပြားနေပါက သင်၏ shell သည် နှေးကွေးလာနိုင်ပါသည်။ သင်သည် ၎င်း၏ စွမ်းဆောင်ရည်ကို အမြဲတမ်း စစ်ဆေးကြည့်ရှုနိုင်ပြီး (profile လုပ်နိုင်ပြီး) မကြာခဏ မသုံးသော သို့မဟုတ် မြန်နှုန်းထက် ပိုမို အရေးမကြီးသော feature များကို ပိတ်ထားနိုင်ပါသည်။

Terminal Emulator များ နှင့် Multiplexer များ

သင်၏ shell ကို စိတ်ကြိုက်ပြင်ဆင်ခြင်းနှင့်အတူ သင်သုံးစွဲမည့် **terminal emulator** ရွေးချယ်မှုနှင့် ယင်း၏ အပြင်အဆင်များကို ရှာဖွေလေ့လာရန် အချိန်အနည်းငယ် ပေးထိုက်ပါသည်။ ပြင်ပတွင် terminal emulator အများအပြား ရှိကြပါသည် (ဒီမှာ [နှိုင်းယှဉ်ချက်](https://anarc.at/blog/2018-04-12-terminal-emulators-1/) ကို ကြည့်နိုင်ပါသည်)။

သင်သည် သင်၏ terminal တွင် နာရီပေါင်း ရာနှင့်ထောင်နှင့်ချီ၍ အချိန်ကုန်လွန်မည် ဖြစ်သောကြောင့် ယင်း၏ setting များကို လေ့လာစိစစ်ခြင်းသည် အကျိုးရှိစေပါသည်။ Terminal တွင် သင် စိတ်ကြိုက် ပြောင်းလဲလိုနိုင်သည့် အချက်အချို့မှာ အောက်ပါအတိုင်း ဖြစ်သည် -

- ဖောင့် ရွေးချယ်မှု
- Color Scheme (အရောင်အသွေး ရွေးချယ်မှု)
- Keyboard shortcuts (ဖြတ်လမ်းနည်းများ)
- Tab/Pane ထောက်ပံ့မှု
- Scrollback အပြင်အဆင်
- စွမ်းဆောင်ရည် (**Alacritty** (https://github.com/jwilm/alacritty) ကဲ့သို့သော ပိုသစ်သည့် terminal အချို့သည် GPU acceleration ကို ထောက်ပံ့ပေးသည်)

tmux (https://github.com/tmux/tmux) ကဲ့သို့သော **terminal multiplexer များ** အကြောင်းကိုလည်း ထည့်သွင်းပြောကြားထိုက်ပါသည်။ `tmux` သည် shell session အများအပြားကို pane များနှင့် tab များအဖြစ် ခွဲခြားပေးနိုင်ပါသည်။ ထို့ပြင် ယင်းသည် attach နှင့် detach လုပ်ခြင်းများကိုလည်း ထောက်ပံ့ပေးရာ remote server တစ်ခုတွင် အလုပ်လုပ်နေချိန် လက်ရှိ process များကို ရုပ်တန့်သွားမည် စိုးရိမ်စရာမလိုဘဲ shell ကို ဆက်လက် run နေစေလိုသည့်အခါ အလွန် အသုံးဝင်ပါသည် (default အားဖြင့် သင် log out လုပ်လိုက်ပါက

သင်၏ process များ သေဆုံးသွားလေ့ရှိသည်)။ ဤနည်းဖြင့် `tmux` ကို သုံး၍ ရှုပ်ထွေးသော terminal layout များထဲသို့ ဝင်ထွက် အသုံးပြုနိုင်ပါသည်။ Terminal emulator များကဲ့သို့ပင် `tmux` သည်လည်း `~/.tmux.conf` ဖိုင်ကို ပြင်ဆင်ခြင်းဖြင့် အဆင့်မြင့် စိတ်ကြိုက် ပြင်ဆင်မှုများကို ထောက်ပံ့ပေးထားပါသည်။

Command-line utility များ

UNIX အခြေခံ operating system အများစုတွင် default ပါဝင်သော command line utility များသည် သင် ပုံမှန် လုပ်ဆောင်ရန် လိုအပ်သည့် အရာများ၏ ၉၉ ရာခိုင်နှုန်းအတွက် လုံလောက်သည်ထက်ပင် ပိုမိုပါသည်။

နောက်လာမည့် ပုဒ်ခွဲအနည်းငယ်တွင် အလွန်အသုံးများသော shell လုပ်ဆောင်ချက်များအတွက် အသုံးပြုရန် ပိုမိုအဆင်ပြေသည့် အခြားရွေးချယ်စရာ tool များကို ရှင်းပြသွားပါမည်။ ဤ tool အချို့သည် command တွင် ပိုမိုကောင်းမွန်သော တိုးတက်သည့် လုပ်ဆောင်ချက်သစ်များကို ပေါင်းစပ်ပေးထားပြီး အချို့မှာမူ ပိုမို ကောင်းမွန်သော default များဖြင့် ရိုးရှင်းပြီး ပိုမိုနားလည်လွယ်သည့် interface ကို ပေးစွမ်းနိုင်ရန်သာ အဓိက ထားထားကြပါသည်။

`fasd` နှင့် `cd`

တိုးတက်လာသော path expansion နှင့် tab autocomplete များ ရှိနေသည့်တိုင် directory များ ပြောင်းလဲခြင်းသည် အတော်လေး ထပ်ခါတလဲလဲ ဖြစ်လာနိုင်ပါသည်။ **Fasd** (https://github.com/clvv/fasd) (သို့မဟုတ် **autojump** (https://github.com/wting/autojump)) သည် သင် သွားရောက်ခဲ့ဖူးသော မကြာသေးမီကနှင့် မကြာခဏ သွားလေ့ရှိသော folder များကို စောင့်ကြည့်မှတ်သားပြီး fuzzy matching ပြုလုပ်ပေးခြင်းဖြင့် ဤပြဿနာကို ဖြေရှင်းပေးပါသည်။

ထို့ကြောင့် ကျွန်ုပ်သည် `/home/user/awesome_project/code` သို့ သွားရောက်ခဲ့ပါက `z code` ကို runလိုက်လျှင် ယင်းသို့ `cd` ဝင်သွားမည် ဖြစ်သည်။ အကယ်၍ code ဟု အမည်ရသော folder အများအပြား ရှိနေပါက ပိုမိုနီးစပ်သော ကိုက်ညီမှုဖြစ်စေရန် `z awe code` ဟု run ပြီး ကွဲပြားအောင် လုပ်နိုင်ပါသည်။ Autojump နှင့် မတူသည်မှာ fasd သည် `cd` ပြုလုပ်မည့်အစား မကြာခဏ သို့မဟုတ် မကြာသေးမီက သုံးထားသော ဖိုင်များ၊ folder များ သို့မဟုတ် နှစ်ခုစလုံး၏ စာလုံးများကို ဖြန့်ကျက် (expand) ပေးသည့် command များကိုပါ ထောက်ပံ့ပေးထားခြင်း ဖြစ်သည်။

`bat` နှင့် `cat`

`cat` သည် ယင်း၏ လုပ်ငန်းကို ပြီးပြည့်စုံစွာ လုပ်ဆောင်နိုင်သော်လည်း **bat** (https://github.com/sharkdp/bat) သည် syntax highlighting (စာကြောင်း ကာလာရောင်စုံပြသမှု)၊ paging၊ စာကြောင်းနံပါတ်များ နှင့် git integration များကို ပေးစွမ်းခြင်းဖြင့် ပိုမိုကောင်းမွန်အောင် ပြုလုပ်ပေးပါသည်။

`exa` / `ranger` နှင့် `ls`

`ls` သည် ကောင်းမွန်သော command တစ်ခုဖြစ်သော်လည်း raw byte များဖြင့် ဖိုင်ဆိုဒ်ကို ပြသခြင်း ကဲ့သို့သော default အချို့မှာ စိတ်ရှုပ်စရာ ကောင်းနိုင်ပါသည်။ **exa** (https://github.com/ogham/exa) သည် ပိုမို ကောင်းမွန်သော default များကို ပေးစွမ်းထားပါသည်

အကယ်၍ သင်သည် folder အများအပြားကို သွားရောက်ကြည့်ရှုရန် နှင့်/သို့မဟုတ် ဖိုင်အများအပြားကို ကြိုတင်ကြည့်ရှုရန် (preview) လိုအပ်ပါက **ranger** (https://github.com/ranger/ranger) ၏ အလွန်ကောင်းမွန်သော interface ကြောင့် `cd` နှင့် `cat` ထက် များစွာ ပိုမို ထိရောက်နိုင်ပါသည်။ ယင်းသည် စိတ်ကြိုက် ပြင်ဆင်ရန် အလွန် အဆင်ပြေပြီး သေချာ စနစ်တကျ ပြင်ဆင်ထားပါက သင်၏ terminal အတွင်း၌ပင် **ဓာတ်ပုံများကို ကြိုတင်ကြည့်ရှုနိုင်မည်** (https://github.com/ranger/ranger/wiki/Image-Previews) ဖြစ်သည်

`fd` နှင့် `find`

fd (https://github.com/sharkdp/fd) သည် `find` ၏ နေရာတွင် သုံးနိုင်သော ရိုးရှင်း၊ မြန်ဆန်ပြီး အသုံးပြုရ လွယ်ကူသည့် အခြားရွေးချယ်စရာ တစ်ခု ဖြစ်သည်။ `--name` flag ကို အသုံးပြုရခြင်း (သင် ဧဂ ရာခိုင်နှုန်း လုပ်ဆောင်လိုသည့် အရာဖြစ်သည်) ကဲ့သို့သော `find` ၏ default အခြေအနေများကို ပိုမို လွယ်ကူစေပြီး နေ့စဉ် သုံးစွဲရန် အဆင်ပြေစေပါသည်။ ထို့အပြင် ယင်းသည် `git` ကို နားလည်ပြီး default အားဖြင့် သင်၏ `.gitignore` နှင့် `.git` folder ထဲရှိ ဖိုင်များကို ကျော်သွားမည် (skip လုပ်မည်) ဖြစ်သည်။ Default အားဖြင့် အရောင်စုံ ကာလာပြသမှု စနစ်လည်း ပါဝင်ပါသည်။

`rg/fzf` နှင့် `grep`

`grep` သည် ကောင်းမွန်သော tool တစ်ခု ဖြစ်သော်လည်း ဖိုင်အများအပြားကို တစ်ပြိုင်နက်တည်း အထဲထိ ဝင်ရောက် ရှာဖွေလိုပါက (grep လုပ်လိုပါက) ထိုအတွက် ပိုမိုကောင်းမွန်သော tool များ ရှိပါသည်။ **ack** (https://github.com/beyondgrep/ack3), **ag** (https://github.com/ggreer/the_silver_searcher) နှင့် **rg** (https://github.com/BurntSushi/ripgrep) တို့သည် သင်၏ gitignore စည်းမျဉ်းများကို လိုက်နာလျက် လက်ရှိ directory အောက်တွင် regex pattern ဖြင့် အဆင့်ဆင့် ရှာဖွေပေးကြပါသည်။ ၎င်းတို့အားလုံးသည် အလုပ် လုပ်ပုံ ခပ်ဆင်ဆင် ဖြစ်သော်လည်း ကျွန်ုပ်၏ home directory တစ်ခုလုံးကို အလွန် လျင်မြန်စွာ ရှာဖွေနိုင် သည့်အတွက် `rg` ကို ပိုမို သဘောကျပါသည်။

ထို့အတူပင် မိမိကိုယ်ကိုယ် `CMD | grep PATTERN` ကို ထပ်ခါတလဲလဲ ပြုလုပ်နေရသည်ကို တွေ့ရလေ့ ရှိပါသည်။ **fzf** (https://github.com/junegunn/fzf) သည် မည်သည့် command ၏ output ကိုမဆို တိုက်ရိုက် တုံ့ပြန် ဆန်းစစ်၍ (interactively) စစ်ထုတ်နိုင်စေသည့် command line fuzzy finder တစ်ခု ဖြစ်ပါသည်။

`rsync` နှင့် `cp/scp`

`mv` နှင့် `scp` တို့သည် အခြေအနေ အများစုအတွက် ပြီးပြည့်စုံသော်လည်း၊ ဖိုင်ပမာဏ အများအပြားကို ကူးယူ/ရွှေ့ပြောင်းသည့်အခါ၊ ဖိုင်ကြီးများကို ကူးယူသည့်အခါ သို့မဟုတ် ပန်းတိုင် (destination) တွင် ဒေတာ အချို့ ရောက်ရှိပြီး ဖြစ်နေသည့်အခါမျိုးတွင် `rsync` သည် အလွန် ကြီးမားသော တိုးတက်ပြောင်းလဲမှု ဖြစ်ပါသည်။ `rsync` သည် ကူးယူပြီးသား ဖိုင်များကို ကျော်သွားမည် ဖြစ်ပြီး `--partial` flag ပါဝင်ပါက ယခင်က ပြတ်တောက်သွားခဲ့သော ကူးယူမှုကို ပြန်လည် စတင်နိုင်မည် ဖြစ်သည်။

`trash` နှင့် `rm`

`rm` သည် ဖိုင်တစ်ခုကို ဖျက်လိုက်ပါက ပြန်လည် ရယူ၍ မရတော့သည့် သဘောရှိသဖြင့် အန္တရာယ်ရှိသော command ဖြစ်ပါသည်။ သို့သော်လည်း ခေတ်ပေါ် OS များသည် file explorer တွင် တစ်ခုခုကို ဖျက်လိုက်သည့်အခါ ထိုသို့ အလုပ်မလုပ်ဘဲ ပုံမှန် ပို့ပစ်လေ့ရှိသော Trash folder ထဲသို့သာ ရွှေ့ပြောင်းပေးလိုက်ကြခြင်း ဖြစ်သည်။

Trash ကို စီမံခန့်ခွဲပုံသည် OS တစ်ခုနှင့်တစ်ခု မတူညီကြသဖြင့် တစ်ခုတည်းသော CLI utility ဟု မရှိပါ။ macOS တွင် **trash** (https://hasseg.org/trash/) ရှိပြီး linux တွင် **trash-cli** (https://github.com/andreafrancia/trash-cli/) စသည်ဖြင့် အသီးသီး ရှိကြပါသည်။

`mosh` နှင့် `ssh`

`ssh` သည် အလွန် အသုံးဝင်သော tool တစ်ခု ဖြစ်သော်လည်း သင်၏ လိုင်းဆက်ကြောင်း နေးကွေးပါက lag ဖြစ်မှုသည် စိတ်ပျက်စရာ ကောင်းလာနိုင်ပြီး ချိတ်ဆက်မှု ပြတ်တောက်သွားပါက ပြန်လည် ချိတ်ဆက်ရလေ့ ရှိသည်။ **mosh** (https://mosh.org/) သည် roaming လုပ်ခြင်းကို ခွင့်ပြုပြီး ပြတ်တောင်းပြတ်တောင်း ချိတ်ဆက်မှုကို ထောက်ပံ့ပေးသည့်အပြင် ဉာဏ်ရည်ထက်မြက်သော local echo ကိုလည်း ပေးစွမ်းနိုင်သည့် အသုံးဝင်သော tool တစ်ခု ဖြစ်ပါသည်။

`tldr` နှင့် `man`

အများစုတွင် command တစ်ခု အလုပ်လုပ်ပုံနှင့် ယင်း၌ ပါဝင်သည့် option များကို `man` သို့မဟုတ် `-h` / `--help` flag များကို သုံး၍ ရှာဖွေနိုင်ပါသည်။ သို့သော်လည်း အချို့သော အခြေအနေများတွင် ၎င်းတို့သည် အသေးစိတ် အလွန်များပြားပါက လိုက်လံ ကြည့်ရှုရသည်မှာ အနည်းငယ် လက်ဝင် စိတ်ပျက်စရာ ဖြစ်နိုင်ပါသည်။

tldr (https://github.com/tldr-pages/tldr) command သည် အသိုက်အဝန်း (community) အခြေပြု documentation စနစ်တစ်ခု ဖြစ်ပြီး command line မှ ရယူသုံးစွဲနိုင်ကာ၊ command အလုပ်လုပ်ပုံနှင့် အသုံးအများဆုံး argument option များအတွက် ရှင်းလင်းသော ဥပမာအနည်းငယ်ကို ပေးစွမ်းထားပါသည်။

`aunpack` နှင့် `tar/unzip/unrar`

ဤ xkcd (https://xkcd.com/1168/) တွင် ညွှန်းဆိုထားသကဲ့သို့ `tar` အတွက် option များကို မှတ်မိရန် ခက်ခဲနိုင်ပြီး တစ်ခါတစ်ရံတွင် .rar ဖိုင်များအတွက် `unrar` ကဲ့သို့သော လုံးဝ မတူညီသည့် အခြား tool တစ်ခုကို အသုံးပြုရန် လိုအပ်တတ်ပါသည်။ atool (https://www.nongnu.org/atool/) package သည် မှန်ကန်သော option များကို အလိုအလျောက် တွက်ချက်ပေးပြီး ဖြည်ထုတ်လိုက်သော archive များကို folder အသစ်တစ်ခု အတွင်းသို့ အမြဲတမ်း ထည့်သွင်းပေးသည့် `aunpack` command ကို ထောက်ပံ့ပေးထားပါသည်။

လေ့ကျင့်ခန်းများ

1. အသုံးအများဆုံး command ၁၀ ခုကို ရယူရန် `cat .bash_history | sort | uniq -c | sort -rn | head -n 10` (သို့မဟုတ် zsh အတွက် `cat .zhistory | sort | uniq -c | sort -rn | head -n 10`) ကို run ပါ။ ပြီးလျှင် ၎င်းတို့အတွက် ပိုမိုတိုတောင်းသော alias များကို ရေးသားရန် စဉ်းစားကြည့်ပါ။
2. Terminal emulator တစ်ခုကို ရွေးချယ်ပြီး အောက်ပါ property များကို မည်သို့ ပြောင်းလဲရမည်ကို ရှာဖွေကြည့်ပါ -

```
- ဖောင့် ရွေးချယ်မှု
- Color scheme။ စံနှုန်းသတ်မှတ်ထားသော scheme တစ်ခုတွင် အရောင် မည်မျှ ပါဝင်သနည်း။ အဘယ်ကြောင့်နည်း။
- Scrollback history ဆိုဒ်
```

1. `fasd` သို့မဟုတ် အလားတူ ဆော့ဖ်ဝဲလ်တစ်ခုခုကို တပ်ဆင်ပါ၊ ပေးပို့လိုက်သော argument များအပေါ် fuzzy matching ပြုလုပ်ပြီး အကောင်းဆုံး ကိုက်ညီသည့် ရလဒ်ကို သင် ရွေးချယ်ထားသော editor တွင် ဖွင့်ပေးသည့် `v` ဟု အမည်ရသော bash/zsh function တစ်ခုကို ရေးသားပါ။ ထို့နောက် ကိုက်ညီမှု အများအပြား ရှိနေပါက `fzf` ဖြင့် ရွေးချယ်နိုင်စေရန် ပြင်ဆင်ပါ။
2. `fzf` သည် fuzzy ရှာဖွေမှုများ ပြုလုပ်ရန် အလွန် အဆင်ပြေပြီး shell history သည် ထိုကဲ့သို့သော ရှာဖွေမှုမျိုးနှင့် သင့်တော်သောကြောင့် `fzf` ကို `^R` သို့ မည်သို့ bind ပြုလုပ်ရမည်ကို စုံစမ်းလေ့လာပါ။ အချက်အလက်အချို့ကို ဒီနေရာတွင် (https://github.com/junegunn/fzf/wiki/Configuring-shell-key-bindings) ရှာဖွေတွေ့ရှိနိုင်ပါသည်။
3. `ack` တွင် `--bar` option သည် မည်သည့်အရာကို လုပ်ဆောင်သနည်း။

🔗 External Links & References (ပြင်ပ လင့်ခ်များနှင့် ကိုးကားချက်များ)

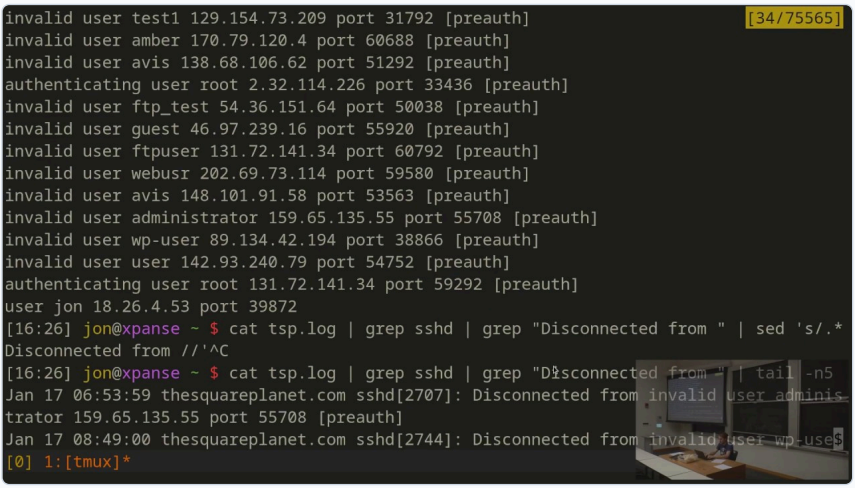
Resource / Description	URL	Micro QR
1. prezto	https://github.com/sorin-ionescu/prezto	
2. oh-my-zsh	https://github.com/robbyrussell/oh-my-zsh	
3. zsh-syntax-highlighting	https://github.com/zsh-users/zsh-syntax-highlighting	
4. zsh-history-substring-search	https://github.com/zsh-users/zsh-history-substring-search	
5. fish	https://fishshell.com/	
6. နှိုင်းယှဉ်ချက်	https://anarc.at/blog/2018-04-12-terminal-emulators-1/	
7. Alacritty	https://github.com/jwilm/alacritty	
8. tmux	https://github.com/tmux/tmux	
9. Fasd	https://github.com/clvv/fasd	
10. autojump	https://github.com/wting/autojump	
11. bat	https://github.com/sharkdp/bat	
12. exa	https://github.com/ogham/exa	
13. ranger	https://github.com/ranger/ranger	
14. ဓာတ်ပုံများကို ကြိုတင်ကြည့်ရှုနိုင်မည်	https://github.com/ranger/ranger/wiki/Image-Previews	
15. fd	https://github.com/sharkdp/fd	

Resource / Description	URL	Micro QR
16. ack	https://github.com/beyondgrep/ack3	
17. ag	https://github.com/ggreer/the_silver_searcher	
18. rg	https://github.com/BurntSushi/ripgrep	
19. fzf	https://github.com/junegunn/fzf	
20. trash	https://hasseg.org/trash/	
21. trash-cli	https://github.com/andreafrancia/trash-cli/	
22. mosh	https://mosh.org/	
23. tldr	https://github.com/tldr-pages/tldr	
24. ဤ xkcd	https://xkcd.com/1168/	
25. atool	https://www.nongnu.org/atool/	
26. ဒီနေရာတွင်	https://github.com/junegunn/fzf/wiki/Configuring-shell-key-bindings	

ဒေတာ ပြုပြင်စီမံခြင်း (Data Wrangling)

▶ LECTURE VIDEO REFERENCE

ဒေတာ ပြုပြင်စီမံခြင်း (Data Wrangling)



Scan the QR code below using your mobile camera or click the link to watch this full lecture online:

Direct Online Link:

https://www.youtube.com/watch?v=VW2jn9Okjhw

သင့်ထံတွင် စာသား (text) များစွာရှိပြီး ယင်းတို့ဖြင့် တစ်စုံတစ်ခု ပြုလုပ်လိုသည့် အခြေအနေမျိုး ကြုံတွေ့ဖူးပါသလား။ အဆင်ပြေပါသည်။ ထိုသို့ ပြုလုပ်ခြင်းသည် Data Wrangling (ဒေတာ ပြုပြင်ပြင်ဆင်ခြင်း) ၏ အဓိက သဘောတရားပင် ဖြစ်ပါသည်။ အတိအကျ ပြောရလျှင် ဒေတာများကို မိမိ လိုချင်သော ပုံစံအတိုင်း အတိအကျ ရရှိသည်အထိ Format တစ်ခုမှ အခြား Format တစ်ခုသို့ ပြောင်းလဲ ပြုပြင်ခြင်း ဖြစ်ပါသည်။

ကျွန်ုပ်တို့သည် အခြေခံ Data Wrangling ကို တွေ့မြင်ခဲ့ကြပြီး ဖြစ်သည်- `journalctl | grep -i intel` ။ - Intel ဟု ပါဝင်သော စနစ် Log ထည့်သွင်းချက်များ အားလုံးကို ရှာဖွေပေးသည် (စာလုံးကြီး/စာလုံးသေး မခွဲခြားပါ - case-insensitive) - အမှန်တော့ Data Wrangling ၏ အဓိက အစိတ်အပိုင်းမှာ မိမိထံတွင် မည်သည့် tool များ ရှိသည်နှင့် ယင်းတို့ကို မည်သို့ ပေါင်းစပ် အသုံးပြုရမည်ကို သိရှိထားခြင်း ဖြစ်ပါသည်။

အစမှ စတင်ကြည့်ကြပါစို့- ကျွန်ုပ်တို့တွင် ဒေတာ အရင်းအမြစ် (data source) တစ်ခုနှင့် ယင်းဒေတာဖြင့် ပြုလုပ်လိုသည့် အရာတစ်ခု လိုအပ်ပါသည်။ Log ဖိုင်များသည် အလွန် ကောင်းမွန်သော စံနမူနာ ဖြစ်ပါသည်၊ အကြောင်းမှာ Log များအကြောင်းကို လေ့လာ စုံစမ်းရန် လိုအပ်သော်လည်း ဖိုင်တစ်ခုလုံးကို ဖတ်ရှုရန် မဖြစ်နိုင်သောကြောင့် ဖြစ်ပါသည်။ အောက်ပါ Server log ကို ကြည့်၍ ကျွန်ုပ်၏ Server ထံသို့ မည်သူက Log in ဝင်ရန် ကြိုးစားနေသည်ကို လေ့လာကြည့်ကြပါစို့-

```
ssh myserver journalctl
```

ဤသည်မှာ အချက်အလက်များ လွန်စွာ များပြားလှပါသည်။ သို့ဖြစ်၍ SSH နှင့် သက်ဆိုင်သည်များကိုသာ ခွဲထုတ်ကြည့်ကြမည်-

```
ssh myserver journalctl | grep sshd
```

ဒီနေရာတွင် Pipe ကို အသုံးပြု၍ *Remote* (အဝေးထိန်း) ဖိုင်ကို မိမိ စက်ပေါ်ရှိ `grep` ထံသို့ Stream ပြုလုပ်ပေးပို့ထားသည်ကို သတိပြုပါ။ `ssh` သည် အလွန် အသုံးဝင်ဆန်းကြယ်လှပါသည်။ သို့သော် ဤသည်မှာလည်း ကျွန်ုပ်တို့ လိုချင်သည်ထက် အချက်အလက်များ များပြားနေပါသေးသည်။ ထို့အပြင် ဖတ်ရှုရန်လည်း ခက်ခဲပါသည်။ ပိုမို ကောင်းမွန်အောင် ပြုလုပ်ကြပါစို့-

```
ssh myserver journalctl | grep sshd | grep "Disconnected from"
```

ဤနေရာတွင် မလိုအပ်သော အချက်အလက်များ (noise) များစွာ ပါဝင်နေသေးသည်။ ယင်းတို့ကို ဖယ်ရှားရန် နည်းလမ်းများ စွာ ရှိသော်လည်း သင်၏ Toolkit ထဲမှ စွမ်းအားအရှိဆုံး Tool တွေထဲက တစ်ခုဖြစ်သည့် `sed` ကို လေ့လာကြည့်ကြပါစို့။

`sed` သည် ရှေးယခင် `ed` editor ပေါ်တွင် အခြေခံ၍ ပြုလုပ်ထားသော “Stream Editor”[9] တစ်ခု ဖြစ်ပါသည်။ ၎င်းတွင် ဖိုင်၏ အကြောင်းအရာများကို တိုက်ရိုက် ပြင်ဆင်ခြင်းထက် (တိုက်ရိုက် ပြင်ဆင်၍လည်း ရသော်လည်း) ဖိုင်ကို မည်သို့ ပြင်ဆင်ရမည်ဆိုသည့် Command တိုများကို ပေးပို့ရခြင်း ဖြစ်ပါသည်။ Command များစွာ ရှိသည့်အနက် အသုံးအများဆုံး တစ်ခုမှာ `s` (Substitution - အစားထိုးခြင်း) ဖြစ်ပါသည်။ ဥပမာ- အောက်ပါအတိုင်း ရေးသားနိုင်ပါသည်-

```
ssh myserver journalctl
  | grep sshd
  | grep "Disconnected from"
  | sed 's/.*Disconnected from //'
```

ယခုလေးတင် ရေးသားလိုက်သည်မှာ ရိုးရှင်းသော *Regular Expression* ဖြစ်ပါသည်။ ၎င်းသည် Pattern များနှင့် စာသားများကို တိုက်ဆိုင်စစ်ဆေး (match) ပေးနိုင်သော စွမ်းအားထက်သည့် နည်းလမ်းတစ်ခု ဖြစ်ပါသည်။ `s` command ကို `s/REGEX/SUBSTITUTION/` ပုံစံဖြင့် ရေးသားပြီး `REGEX` သည် သင်ရှာဖွေလိုသော Regular Expression ဖြစ်ကာ `SUBSTITUTION` သည် ကိုက်ညီသော စာသားနေရာတွင် အစားထိုးလိုသည့် စာသား ဖြစ်ပါသည်။

Regular expressions

Regular expression များသည် အသုံးများပြီး အလွန် အသုံးဝင်သောကြောင့် ၎င်းတို့ မည်သို့ အလုပ်လုပ်သည်ကို အချိန်ပေး၍ နားလည်သဘောပေါက်အောင် လေ့လာသင့်ပါသည်။ အထက်တွင် ကျွန်ုပ်တို့ အသုံးပြုခဲ့သည့် `/.*Disconnected from /` ကို စတင်ကြည့်ကြပါစို့။ Regular expression များကို ပုံမှန်အားဖြင့် (အမြဲတမ်း တော့ မဟုတ်ပါ) `/` ဖြင့် ဝန်းရံလေ့ ရှိကြသည်။ ASCII စာလုံး အများစုသည် ၎င်းတို့၏ ပုံမှန် အဓိပ္ပာယ်အတိုင်း သက်ရောက်သော်လည်း အချို့သော စာလုံးများသည် “အထူး” ကိုက်ညီမှု ပြုမူဆောင်ရွက်ချက်များ (special matching behavior) ရှိကြသည်။ မည်သည့် စာလုံးက မည်သို့ ပြုလုပ်သည်ဆိုသည်မှာ Regular Expression တည်ဆောက်ပုံ အကောင်အထည်ဖော်မှု (implementation) များအပေါ် မူတည်၍ အနည်းငယ် ကွဲပြားကြပြီး ယင်းသည် အလွန် ရှုပ်ထွေးစိတ်ပျက်စရာ ကောင်းလှပါသည်။ အသုံးများသော Pattern အချို့မှာ အောက်ပါအတိုင်း ဖြစ်သည်-

- `.` သည် Newline မှအပါအဝင် “မည်သည့် စာလုံးတစ်လုံးမဆို” ကို အဓိပ္ပာယ်ရသည်
- `*` ရှေ့မှ စာလုံး သို့မဟုတ် Pattern ဝ ခု သို့မဟုတ် ဝ ခုထက်ပို၍ ပါဝင်ခြင်း
- `+` ရှေ့မှ စာလုံး သို့မဟုတ် Pattern ၁ ခု သို့မဟုတ် ၁ ခုထက်ပို၍ ပါဝင်ခြင်း
- `[abc]` `a` ၊ `b` သို့မဟုတ် `c` စာလုံးများအနက် မည်သည့် စာလုံးတစ်လုံးမဆို
- `(RX1|RX2)` `RX1` သို့မဟုတ် `RX2` နှင့် ကိုက်ညီသည့် အရာတစ်ခုခု
- `^` စာကြောင်း၏ အစ
- `$` စာကြောင်း၏ အဆုံး

`sed` ၏ Regular Expression များသည် အနည်းငယ် ထူးဆန်းပြီး အထူး အဓိပ္ပာယ် သက်ရောက်စေရန် ၎င်းတို့၏ ရှေ့တွင် `\` ထည့်သွင်းပေးရန် လိုအပ်ပါသည်။ သို့မဟုတ်ပါက `-E` flag ကို သုံးနိုင်ပါသည်။

ထို့ကြောင့် `/.*Disconnected from /` ကို ပြန်လည်ကြည့်ရှုပါက ၎င်းသည် မည်သည့် စာလုံး အရေအတွက်မဆို ပါဝင်ပြီး ၎င်းနောက်တွင် “Disconnected from” ဟူသည့် စာသား အတိအကျ ပါဝင်သော မည်သည့် စာသားမဆိုနှင့် ကိုက်ညီမှု ရှိသည်ကို တွေ့ရမည် ဖြစ်ပါသည်။ ဤသည်မှာ ကျွန်ုပ်တို့ လိုချင်သည့် အရာ ဖြစ်သည်။ သို့သော် သတိပြုပါ၊ Regular Expression များသည် ရှုပ်ထွေးလှပါသည်။ အကယ်၍ တစ်စုံတစ်ယောက်က “Disconnected from” ဟူသော Username ဖြင့် Log in ဝင်ရန် ကြိုးစားခဲ့ပါက မည်သို့ ဖြစ်မည်နည်း။ အောက်ပါအတိုင်း ရှိနေမည် ဖြစ်သည်-

```
Jan 17 03:13:00 thesquareplanet.com sshd[2631]: Disconnected from invalid user
Disconnected from 46.97.239.16 port 55920 [preauth]
```

ကျွန်ုပ်တို့ မည်သည့် ရလဒ် ရရှိမည်နည်း။ အမှန်တော့ `*` နှင့် `+` တို့သည် ပုံမှန်အားဖြင့် “Greedy” (ဝါးမျိုသော/အတတ်နိုင်ဆုံး အများဆုံး ယူသော) သဘောရှိကြသည်။ ၎င်းတို့သည် တတ်နိုင်သမျှ စာသား အရေအတွက် အများဆုံးနှင့် ကိုက်ညီအောင် ပြုလုပ်ကြသည်။ သို့ဖြစ်၍ အထက်ပါ ဥပမာတွင် အောက်ပါ အတိုင်းသာ ကျန်ရစ်မည် ဖြစ်သည်-

```
46.97.239.16 port 55920 [preauth]
```

ယင်းသည် ကျွန်ုပ်တို့ လိုချင်သော ရလဒ် ဟုတ်ချင်မှ ဟုတ်ပေလိမ့်မည်။ Regular Expression Implementation အချို့တွင် `*` သို့မဟုတ် `+` ၏ အနောက်တွင် `?` ထည့်သွင်းခြင်းဖြင့် Non-greedy (အနည်းဆုံးသာ ယူသော) ပုံစံသို့ ပြောင်းလဲနိုင်သော်လည်း စိတ်ပျက်စရာကောင်းသည်မှာ `sed` က ယင်းကို မပံ့ပိုးပါ။ သို့သော် ယင်းကို ပံ့ပိုးပေးသည့် Perl ၏ Command-line mode သို့ ပြောင်းလဲ အသုံးပြုနိုင်ပါသည်-

```
perl -pe 's/.*?Disconnected from //'
```

သို့သော် ဤကဲ့သို့သော လုပ်ဆောင်ချက်များအတွက် `sed` သည် ပိုမို အသုံးများသော Tool ဖြစ်သောကြောင့် ကျန်ရှိသည့် အပိုင်းများတွင် `sed` ကိုသာ ဆက်လက် အသုံးပြုသွားမည် ဖြစ်ပါသည်။ `sed` သည် ကိုက်ညီသော စာကြောင်းများ၏ နောက်ဆက်တွဲ စာကြောင်းများကို ရိုက်နှိပ်ခြင်း၊ Command တစ်ခုတည်းတွင် အစားထိုးမှု အများအပြား ပြုလုပ်ခြင်း၊ အရာဝတ္ထုများ ရှာဖွေခြင်း အစရှိသော အသုံးဝင်သည့် အခြား လုပ်ဆောင်ချက်များကိုလည်း ပြုလုပ်နိုင်ပါသည်။ သို့သော် ဤနေရာတွင် ယင်းတို့ကို အသေးစိတ် ဖော်ပြသွားမည် မဟုတ်ပါ။ `sed` သည် သီးခြား သီးသန့် လေ့လာရမည့် ခေါင်းစဉ်တစ်ခု ဖြစ်သော်လည်း ၎င်းထက် ပိုမို ကောင်းမွန်သည့် Tool များလည်း ရှိလေ့ ရှိပါသည်။

ကောင်းပါပြီ၊ ထို့ကြောင့် ကျွန်ုပ်တို့ထံတွင် ဖယ်ရှားလိုသော Suffix (အနောက်ဆက်) တစ်ခုလည်း ရှိနေပါသည်။ ၎င်းကို မည်သို့ ဖယ်ရှားနိုင်မည်နည်း။ အထူးသဖြင့် Username တွင် space များ ပါဝင်နိုင်

သည့်အတွက် Username ၏ အနောက်မှ စာသားများကိုသာ သီးသန့် တိုက်ဆိုင်စစ်ဆေးရန် အနည်းငယ် ခက်ခဲပါသည်။ ကျွန်ုပ်တို့ ပြုလုပ်ရန် လိုအပ်သည်မှာ စာကြောင်း *တစ်ကြောင်းလုံး* ကို တိုက်ဆိုင်စစ်ဆေးရန် ဖြစ်သည်-

```
 | sed -E 's/.*Disconnected from (invalid |authenticating )?user .* [^ ]+ port
[0-9]+( \[preauth\])?$//'
```

regex debugger (https://regex101.com/r/qqbZqh/2) ဖြင့် မည်သို့ အလုပ်လုပ်သည်ကို ကြည့်ရှုကြပါစို့။ အစပိုင်းသည် ယခင်အတိုင်းပင် ဖြစ်ပါသည်။ ထို့နောက် “user” ရှေ့ဆက် အမျိုးအစားများ (Log များတွင် ရှေ့ဆက် ပုံစံ နှစ်ခု ရှိသည်) နှင့် တိုက်ဆိုင် စစ်ဆေးပါသည်။ ထို့နောက် Username ရှိသည့် စာလုံး တန်းစီမှုနှင့် တိုက်ဆိုင် စစ်ဆေးပါသည်။ ထို့နောက် စကားလုံး တစ်လုံးတည်း (`[^ ]+` ; space မဟုတ်သော စာလုံးများ၏ သဲသဲမဲမဲ အစီအစဉ်) နှင့် တိုက်ဆိုင် စစ်ဆေးပါသည်။ ထို့နောက် “port” စကားလုံးနှင့် ယင်းနောက်တွင် ဂဏန်းစဉ် အစီအစဉ် ပါဝင်ပါသည်။ ထို့နောက် ဖြစ်နိုင်ဖွယ်ရှိသော `[preauth]` အနောက်ဆက် ပါဝင်ပြီး စာကြောင်း၏ အဆုံး သို့ ရောက်ရှိပါသည်။

ဤနည်းလမ်းကို အသုံးပြုခြင်းဖြင့် “Disconnected from” ဟူသော Username သည် ကျွန်ုပ်တို့အား ထပ်မံ ရှုပ်ထွေးစေမည် မဟုတ်သည်ကို သတိပြုပါ၊ အဘယ်ကြောင့်နည်းဆိုသည်ကို မြင်တွေ့နိုင်ပါသလား။

သို့သော် ဤနေရာတွင် ပြဿနာတစ်ခု ရှိသည်၊ ယင်းမှာ Log စာကြောင်း တစ်ခုလုံး ကွက်လပ် ဖြစ်သွားခြင်း ဖြစ်ပါသည်။ အမှန်တော့ ကျွန်ုပ်တို့သည် Username ကို *သိမ်းဆည်းထားလိုခြင်း* ဖြစ်သည်။ ဤအတွက် “Capture Groups” များကို အသုံးပြုနိုင်ပါသည်။ ကွင်းစကွင်းပိတ် (parentheses) များဖြင့် ဝန်းရံထားသော Regex ဖြင့် တိုက်ဆိုင်စစ်ဆေးမိသော မည်သည့် စာသားမဆိုကို နံပါတ်တပ်ထားသော Capture Group အဖြစ် သိမ်းဆည်းပေးပါသည်။ ယင်းတို့ကို အစားထိုးခြင်း နေရာတွင် `\1` ၊ `\2` ၊ `\3` အစရှိသည်ဖြင့် ရရှိ အသုံးပြုနိုင်ပါသည် (အချို့ Engine များတွင် Pattern အတွင်း၌ပင် အသုံးပြုနိုင်သည်)။ သို့ဖြစ်၍-

```
 | sed -E 's/.*Disconnected from (invalid |authenticating )?user (.*) [^ ]+ por
t [0-9]+( \[preauth\])?$/\2/'
```

သင် ခန့်မှန်းမိသည့်အတိုင်း လွန်စွာ ရှုပ်ထွေးသော Regular Expression များကို ရေးသားနိုင်ပါသည်။ ဥပမာ အားဖြင့် **e-mail address** (https://www.regular-expressions.info/email.html) များကို မည်သို့ တိုက်ဆိုင်စစ်ဆေးနိုင်ကြောင်း ရေးသားထားသော ဆောင်းပါးကို ကြည့်ပါ။ ယင်းမှာ **မလွယ်ကူပါ** (https://web.archive.org/web/20221223174323/http://emailregex.com/)။ **ဆွေးနွေးငြင်းခုံမှုများစွာ** (https://stackoverflow.com/questions/201323/how-to-validate-an-email-address-using-a-regular-expression/1917982) လည်း ရှိပါသည်။ ထို့အပြင် **Test များ ရေးသားခြင်း** (https://fightingforalostcause.net/content/misc/2006/compare-email-regex.php)၊ **Test matrices များ ပြုလုပ်ခြင်း** (https://mathiasbynens.be/demo/url-regex) စသည်တို့ ရှိကြပါသည်။

ပေးထားသော ကိန်းဂဏန်းတစ်ခုသည် သဘာဝကိန်း (Prime number) ဟုတ်မဟုတ် (https://www.noulakaz.net/2007/03/18/a-regular-expression-to-check-for-prime-numbers/) စစ်ဆေးနိုင်သည့် Regex ကိုပင် ရေးသားနိုင်ပါသည်။

Regular expression များသည် အမှန်အတိုင်း ရရှိအောင် ပြုလုပ်ရန် ခက်ခဲလွန်းသည်ဟု နာမည်ကြီး သော်လည်း သင်၏ Toolkit တွင် ရှိထားပါက အလွန် အသုံးဝင်သော နည်းလမ်း ဖြစ်ပါသည်။

Back to data wrangling

ကောင်းပြီ၊ ထို့ကြောင့် ယခု ကျွန်ုပ်တို့ထံတွင် အောက်ပါအတိုင်း ရရှိထားသည်-

```
ssh myserver journalctl
 | grep sshd
 | grep "Disconnected from"
 | sed -E 's/.*Disconnected from (invalid |authenticating )?user (.*) [^ ]+ por
t [0-9]+( \[preauth\])?$/\2/'
```

ကျွန်ုပ်တို့သည် `sed` တစ်ခုတည်းဖြင့်လည်း ပြုလုပ်နိုင်ပါသည်၊ သို့သော် အဘယ်ကြောင့် ပြုလုပ်မည်နည်း။ ပျော်စရာ ကောင်းသောကြောင့် ဖြစ်ပါသည်။

```
ssh myserver journalctl
 | sed -E
   -e '/Disconnected from/!d'
   -e 's/.*Disconnected from (invalid |authenticating )?user (.*) [^ ]+ port [0
-9]+( \[preauth\])?$/\2/'
```

ဤသည်မှာ `sed` ၏ စွမ်းဆောင်ရည် အချို့ကို ဖော်ပြနေခြင်း ဖြစ်ပါသည်။ `sed` သည် စာသားများ ထည့်သွင်းခြင်း (`i` command ဖြင့်)၊ စာကြောင်းများကို အတိအကျ ထုတ်ပြခြင်း (`p` command ဖြင့်)၊ အညွှန်း (index) အလိုက် စာကြောင်းများ ရွေးချယ်ခြင်းနှင့် အခြား အရာများစွာကို ပြုလုပ်နိုင်ပါသည်။ `man sed` တွင် ကြည့်ရှုပါ!

မည်သို့ပင်ဖြစ်စေ၊ ယခု ကျွန်ုပ်တို့ ရရှိထားသည်မှာ Log in ဝင်ရန် ကြိုးပမ်းခဲ့သည့် Username များ အားလုံး၏ စာရင်း ဖြစ်ပါသည်။ သို့သော် ဤသည်မှာ သိပ်ပြီး အသုံးမဝင်လှပါ။ အသုံးအများဆုံး Username များကို ရှာဖွေကြည့်ကြပါစို့-

```
ssh myserver journalctl
 | grep sshd
 | grep "Disconnected from"
 | sed -E 's/.*Disconnected from (invalid |authenticating )?user (.*) [^ ]+ por
t [0-9]+( \[preauth\])?$/\2/'
 | sort | uniq -c
```

`sort` သည် ယင်း၏ Input ကို စီပေးမည် ဖြစ်ပါသည်။ `uniq -c` သည် တူညီသော ဆက်တိုက် စာကြောင်းများကို စာကြောင်း တစ်ကြောင်းတည်းအဖြစ် ပေါင်းစည်းပေးပြီး ၎င်းတို့ ပါဝင်သည့် အကြိမ်အရေအတွက် Count ကို ရှေ့မှ ထည့်သွင်းပေးသည်။ ယင်းကိုလည်း ထပ်မံ Sort ပြုလုပ်ပြီး အသုံးအများဆုံး Login များကိုသာ ချန်ထားလိုပေလိမ့်မည်-

```
ssh myserver journalctl
 | grep sshd
 | grep "Disconnected from"
 | sed -E 's/.*Disconnected from (invalid |authenticating )?user (.*) [^ ]+ por
t [0-9]+( \[preauth\])?$/\2/'
 | sort | uniq -c
 | sort -nk1,1 | tail -n10
```

`sort -n` သည် (အက္ခရာစဉ် မဟုတ်ဘဲ) ကိန်းဂဏန်း အစဉ်အတိုင်း စီပေးမည် ဖြစ်သည်။ `-k1,1` ၏ အဓိပ္ပာယ်မှာ “Whitespace ဖြင့် ခွဲခြားထားသော ပထမဆုံး Column ကိုသာ စီရန်” ဖြစ်ပါသည်။ `,n` အပိုင်းသည် `n` ခုမြောက် Field အထိ စီရန် ပြောခြင်းဖြစ်ပြီး ပုံမှန်အတိုင်းဆိုလျှင် စာကြောင်း၏ အဆုံးအထိ ဖြစ်ပါသည်။ ဤ *သီးသန့် ဥပမာတွင်* စာကြောင်း တစ်ကြောင်းလုံးအလိုက် စီခြင်းသည် ကွဲပြားမှု ရှိမည် မဟုတ်သော်လည်း ကျွန်ုပ်တို့သည် သင်ယူရန် ဤနေရာသို့ ရောက်ရှိနေခြင်း ဖြစ်ပါသည်။

အကယ်၍ ကျွန်ုပ်တို့သည် အသုံး *အနည်းဆုံး* အရာများကို လိုချင်ပါက `tail` အစား `head` ကို အသုံးပြုနိုင်ပါသည်။ ပြောင်းပြန် အစီအစဉ်အတိုင်း စီပေးသည့် `sort -r` လည်း ရှိပါသည်။

ကောင်းပြီ၊ ယင်းသည် အလွန် ကောင်းမွန်လှသည်၊ သို့သော် ကျွန်ုပ်တို့သည် Username များကိုသာ ရယူလိုပြီး စာကြောင်းတစ်ကြောင်းလျှင် တစ်ခုစီ မဟုတ်ဘဲ ထုတ်ယူချင်ပါသည်-

```
ssh myserver journalctl
 | grep sshd
 | grep "Disconnected from"
 | sed -E 's/.*Disconnected from (invalid |authenticating )?user (.*) [^ ]+ por
t [0-9]+( \[preauth\])?$/\2/'
 | sort | uniq -c
 | sort -nk1,1 | tail -n10
 | awk '{print $2}' | paste -sd,
```

`paste` ဖြင့် စတင်ကြည့်ကြပါစို့- ၎င်းသည် စာကြောင်းများကို ပေးထားသော စာလုံးတစ်လုံးတည်းဖြစ်သည့် Delimiter (`-d`) ဖြင့် ပေါင်းစည်းပေးပါသည် (`-s`)။ သို့သော် ဤ `awk` ဆိုသည်မှာ မည်သည့်အရာ နည်း။

awk – another editor

`awk` သည် စာသား Stream များကို ပြုပြင်ဆောင်ရွက်ရာတွင် လွန်စွာ ကျွမ်းကျင်သည့် Programming ဓာတ် ခံရှိသော ဘာသာစကားတစ်ခု ဖြစ်ပါသည်။ အကယ်၍ `awk` ကို သေချာစွာ သင်ယူမည်ဆိုပါက ပြောစရာ များ စွာ ရှိသော်လည်း ဤနေရာတွင် အခြား အရာများကဲ့သို့ပင် အခြေခံများကိုသာ သွားရောက်မည် ဖြစ် ပါသည်။

ပထမဦးစွာ `{print $2}` က မည်သည်ကို ပြုလုပ်သနည်း။ အမှန်တော့ `awk` ပရိုဂရမ်များသည် ရွေးချယ် နိုင်သော (optional) Pattern တစ်ခု အပြင် ပေးထားသော စာကြောင်းနှင့် ကိုက်ညီပါက မည်သို့ ပြုလုပ်ရမည် ကို ဖော်ပြသည့် Block တစ်ခု ပုံစံ ယူကြသည်။ ပုံသေ Pattern (ကျွန်ုပ်တို့ အထက်တွင် အသုံးပြုခဲ့သည့် အတိုင်း) သည် စာကြောင်း အားလုံးနှင့် ကိုက်ညီပါသည်။ Block အတွင်းတွင် `$0` ကို စာကြောင်း တစ် ကြောင်းလုံး၏ အကြောင်းအရာအဖြစ် သတ်မှတ်ပြီး `$1` မှ `$n` အထိကို `awk` field separator (ပုံမှန် အားဖြင့် whitespace၊ `-F` ဖြင့် ပြောင်းလဲနိုင်သည်) ဖြင့် ခွဲခြားထားသော စာကြောင်း၏ `n` ခုမြောက် *field* အဖြစ် သတ်မှတ်ပါသည်။ ဤနေရာတွင် စာကြောင်းတိုင်းအတွက် ဒုတိယမြောက် Field ၏ အကြောင်းအရာ ကို ရိုက်နှိပ်ရန် ပြောထားခြင်းဖြစ်ပြီး ယင်းမှာ Username ဖြစ်နေပါသည်!

ပိုမို ဆန်းသစ်သော အရာတစ်ခုခုကို ပြုလုပ်နိုင် မနိုင် ကြည့်ကြပါစို့။ `c` ဖြင့် စပြီး `e` ဖြင့် ဆုံးသည့် တစ် ကြိမ်သာ သုံးထားသော Username အရေအတွက်ကို တွက်ချက်ကြပါစို့-

```
 | awk '$1 == 1 && $2 ~ /^c[^ ]*e$/ { print $2 }' | wc -l
```

ဤနေရာတွင် ရှင်းလင်းစရာများစွာ ရှိပါသည်။ ပထမဦးစွာ ကျွန်ုပ်တို့ထံတွင် Pattern တစ်ခု ရှိလာသည်ကို သတိပြုပါ (`{...}` မတိုင်မီ ပါဝင်သော အရာများ)။ ယင်း Pattern က စာကြောင်း၏ ပထမ Field သည် 1 နှင့် ညီမျှရမည်ဖြစ်ကြောင်း (`uniq -c` မှ ပေးသော အရေအတွက် ဖြစ်သည်)၊ ထို့အပြင် ဒုတိယ Field သည်

ပေးထားသော Regular Expression နှင့် ကိုက်ညီရမည် ဖြစ်ကြောင်း ဖော်ပြထားသည်။ Block ကမူ Username ကို ထုတ်ပြရန်သာ ပြောထားပါသည်။ ထို့နောက် Output စာကြောင်း အရေအတွက်ကို `wc -l` ဖြင့် ရေတွက်လိုက်ပါသည်။

သို့သော် `awk` သည် Programming ဘာသာစကားတစ်ခု ဖြစ်သည်ကို မှတ်မိပါသလား။

```
BEGIN { rows = 0 }
$1 == 1 && $2 ~ /^c[^ ]*e$/ { rows += $1 }
END { print rows }
```

`BEGIN` သည် Input ၏ အစနှင့် ကိုက်ညီသော Pattern ဖြစ်သည် (ထို့အပြင် `END` သည် အဆုံးနှင့် ကိုက်ညီသည်)။ ယခု စာကြောင်းအလိုက် Block သည် ပထမ Field မှ အရေအတွက်ကို ပေါင်းစပ်သွားရုံသာ ဖြစ်ပြီး (ဤနေရာတွင် အမြဲ 1 ဖြစ်နေမည် ဖြစ်သော်လည်း) ထို့နောက် အဆုံးတွင် ထုတ်ပြပေးမည် ဖြစ်ပါသည်။ အမှန်တော့ `awk` က [အရာအားလုံးကို ပြုလုပ်နိုင်သောကြောင့်](https://web.archive.org/web/20251210045942/https://backreference.org/2010/02/10/idiomatic-awk/) `grep` နှင့် `sed` တို့ကို လုံးဝ ဖယ်ရှားပစ်နိုင်သော်လည်း ယင်းကို ဖတ်ရှုသူများအတွက် လေ့ကျင့်ခန်းအဖြစ် ချန်လှပ်ထားခဲ့ပါမည်။

Analyzing data

သင်သည် သန်ချာ တွက်ချက်မှုများကို ပြုလုပ်နိုင်ပါသည်!

```
  | paste -sd+ | bc -l
```

```
echo "2*($(data | paste -sd+))" | bc -l
```

သင်သည် စာရင်းအင်း (stats) များကို နည်းလမ်းအမျိုးမျိုးဖြင့် ရယူနိုင်ပါသည်။ **`st`** (https://github.com/nferraz/st) သည် သပ်ရပ်ကောင်းမွန်လှသော်လည်း အကယ်၍ သင့်ထံတွင် R ရှိပြီးသားဆိုပါက-

```
ssh myserver journalctl
 | grep sshd
 | grep "Disconnected from"
 | sed -E 's/.*Disconnected from (invalid |authenticating )?user (.*) [^ ]+ por
t [0-9]+( \[preauth\])?$/\2/'
 | sort | uniq -c
 | awk '{print $1}' | R --slave -e 'x <- scan(file="stdin", quiet=TRUE); summar
y(x)'
```

R သည် ဒေတာ သုံးသပ်လေ့လာခြင်းနှင့် **Plot တိုင်းတာပုံဆွဲခြင်း** (https://ggplot2.tidyverse.org/) တို့တွင် ကျွမ်းကျင်သော အခြား (ဆန်းကြယ်သည့်) Programming ဘာသာစကားတစ်ခု ဖြစ်ပါသည်။ ကျွန်ုပ်တို့ အသေးစိတ်များစွာ သွားရောက်မည် မဟုတ်သော်လည်း `summary` သည် Matrix တစ်ခုအကြောင်း စာရင်းအင်း အကျဉ်းချုပ်ကို ထုတ်ပြပေးကြောင်း ဖော်ပြပါက လုံလောက်ပါသည်၊ ထို့အပြင် ဂဏန်းများ ပါဝင်သော Input Stream မှ Matrix တစ်ခုကို တွက်ချက်လိုက်သောကြောင့် R သည် ကျွန်ုပ်တို့ လိုချင်သော စာရင်းအင်းများကို ထုတ်ပေးလိုက်ခြင်း ဖြစ်ပါသည်!

အကယ်၍ ရိုးရှင်းသော Plotting အချို့ကိုသာ ပြုလုပ်လိုပါက `gnuplot` ကို အသုံးပြုနိုင်ပါသည်-

```
ssh myserver journalctl
 | grep sshd
 | grep "Disconnected from"
 | sed -E 's/.*Disconnected from (invalid |authenticating )?user (.*) [^ ]+ por
t [0-9]+( \[preauth\])?$/\2/'
 | sort | uniq -c
 | sort -nk1,1 | tail -n10
 | gnuplot -p -e 'set boxwidth 0.5; plot "-" using 1:xtic(2) with boxes'
```

Data wrangling to make arguments

အချို့ အချိန်များတွင် ပိုမို ရှည်လျားသော စာရင်းအပေါ် အခြေခံ၍ ထည့်သွင်းရန် သို့မဟုတ် ဖယ်ရှားရန် အရာများကို ရှာဖွေနိုင်ရန် Data Wrangling ပြုလုပ်လိုကြသည်။ ယခုတိုင်အောင် ဆွေးနွေးခဲ့သော Data Wrangling နည်းလမ်းများ + `xargs` သည် စွမ်းအားထက်မြက်သည့် အတွဲအစပ်တစ်ခု ဖြစ်နိုင်ပါသည်-

```
rustup toolchain list | grep nightly | grep -vE "nightly-x86|01-17" | sed 's/-x
86.*//' | xargs rustup toolchain uninstall
```

လေ့ကျင့်ခန်းများ

1. အကယ်၍ သင်သည် Regular Expression များနှင့် ယဉ်ပါးမှု မရှိသေးပါက ဤနေရာ (https://regexone.com/) တွင် အခြေခံ အများစုကို လွှမ်းခြုံထားသော တိုတောင်းသည့် တိုက်ရိုက် လေ့ကျင့်နိုင်သော Tutorial ရှိပါသည်။

2. `sed s/REGEX/SUBSTITUTION/g` သည် ပုံမှန် sed နှင့် မည်သို့ ကွဲပြားသနည်း။ `/I` သို့မဟုတ် `/m` တို့ကော မည်သို့ ကွဲပြားသနည်း။

3. In-place အစားထိုးမှု ပြုလုပ်ရန်အတွက် `sed s/REGEX/SUBSTITUTION/ input.txt > input.txt` ကဲ့သို့ ပြုလုပ်ရန် လွန်စွာ ဆွဲဆောင်မှု ရှိသည်။ သို့သော် ဤသည်မှာ မကောင်းသော အကြံဉာဏ် ဖြစ်သည်၊ အဘယ်ကြောင့်နည်း။ ဤသည်မှာ `sed` တစ်ခုတည်းအတွက် ဖြစ်ပေါ်သော အရာ ဟုတ်ပါသလား။

4. သင် ယဉ်ပါးသည့် ဘာသာစကားတစ်ခုခုတွင် Regex ကို အသုံးပြု၍ ရိုးရှင်းသော grep နှင့် တူညီသော Tool တစ်ခု အကောင်အထည်ဖော်ပါ။ အကယ်၍ Output ကို grep ကဲ့သို့ အရောင်များဖြင့် ပေါ်လွင်စေလိုပါက ANSI color escape sequence များကို ရှာဖွေပါ။

5. အချို့သော အချိန်များတွင် ဖိုင်အမည်များ ပြောင်းလဲခြင်းကဲ့သို့သော လုပ်ဆောင်ချက်များသည် `mv` ကဲ့သို့သော ရိုးရှင်းသည့် Command များဖြင့် ပြုလုပ်ရန် ရှုပ်ထွေးနိုင်ပါသည်။ `rename` သည် ယင်းကို ပြုလုပ်ရန် အသုံးဝင်သော Tool တစ်ခုဖြစ်ပြီး sed ကဲ့သို့ Syntax ရှိပါသည်။ အမည်တွင် Space ပါသော ဖိုင်များစွာ ဖန်တီးကြည့်ပြီး ယင်းတို့ကို Underscore ဖြင့် အစားထိုးရန် `rename` ကို အသုံးပြုပါ။

6. သင်၏ ပြီးခဲ့သော Reboot သုံးကြိမ်အကြား မျှဝေထားခြင်း *မရှိသော* Boot မက်ဆေ့ဂျ်များကို ရှာဖွေပါ (`journalctl` ၏ `-b` flag ကို ကြည့်ပါ)။ ယင်းက ပိုမို လွယ်ကူစေနိုင်သည့်အတွက် Boot Log အားလုံးကို ဖိုင်တစ်ခုတည်းအဖြစ် ပေါင်းစပ်လိုက်နိုင်ပါသည်။

7. အောက်ပါ မက်ဆေ့ဂျ်များ၏ Log Timestamp ကို အသုံးပြု၍ ပြီးခဲ့သော Reboot ၁၀ ကြိမ်အတွင်း သင်၏ စနစ် Boot တက်ချိန် စာရင်းအင်းများကို ထုတ်ယူပါ-

```
Logs begin at ...
```

နှင့်

```
systemd[577]: Startup finished in ...
```

8. `/usr/share/dict/words` တွင် နည်းဆုံး `a` သုံးခု ပါဝင်ပြီး အဆုံးတွင် `'s` ပါဝင်ခြင်း မရှိသော စကားလုံး အရေအတွက်ကို ရှာပါ။ ယင်း စကားလုံးများ၏ အသုံးအများဆုံး အဆုံးသတ် စာလုံးနှစ်လုံး ပုံစံ သုံးခုမှာ မည်သည်တို့ နည်း။ `sed` ၏ `y` command သို့မဟုတ် `tr` ပရိုဂရမ်သည် စာလုံးကြီး/စာလုံးသေး မခွဲခြားဘဲ စစ်ဆေးရာတွင် ကူညီပေးနိုင်ပါသည်။ ယင်း စကားလုံးနှစ်လုံး ပုံစံ အရေအတွက် မည်မျှ ရှိသနည်း။ ထို့အပြင် ပိုမို စိန်ခေါ်မှုအတွက်- မည်သည့် ပုံစံများ ပါဝင်ခြင်း မရှိပါသနည်း။

9. ဤအရာ (https://commons.wikimedia.org/wiki/Data:Wikipedia_statistics/data.tab) သို့မဟုတ် ဤအရာ (https://ucr.fbi.gov/crime-in-the-u.s/2016/crime-in-the-u.s.-2016/topic-pages/tables/table-1) ကဲ့သို့သော အွန်လိုင်း ဒေတာ အစုအဝေး တစ်ခုကို ရှာပါ။ သို့မဟုတ် ဤနေရာမှ (https://www.springboard.com/blog/data-science/free-public-data-sets-data-science-project/) အခြား ဒေတာ အစုအဝေး တစ်ခုလည်း ဖြစ်နိုင်ပါသည်။ `curl` ကို အသုံးပြု၍ ယင်းကို ရယူပြီး ဂဏန်း ဒေတာ Column နှစ်ခုကိုသာ ထုတ်ယူပါ။ အကယ်၍ HTML ဒေတာကို ရယူနေခြင်းဖြစ်ပါက `pup` (https://github.com/EricChiang/pup) သည် ကူညီပေးနိုင်ပါသည်။ JSON ဒေတာအတွက်မူ `jq` (https://stedolan.github.io/jq/) ကို စမ်းသပ်ကြည့်ပါ။ Command တစ်ခုတည်းဖြင့် Column တစ်ခု၏ အနည်းဆုံးနှင့် အများဆုံး ကိန်းဂဏန်းကို ရှာပါ၊ အခြား Command တစ်ခုဖြင့် Column နှစ်ခုအကြား ခြားနားချက်၏ စုစုပေါင်း ပေါင်းလဒ်ကို ရှာပါ။

🔗 External Links & References (ပြင်ပ လင့်ခ်များနှင့် ကိုးကားချက်များ)

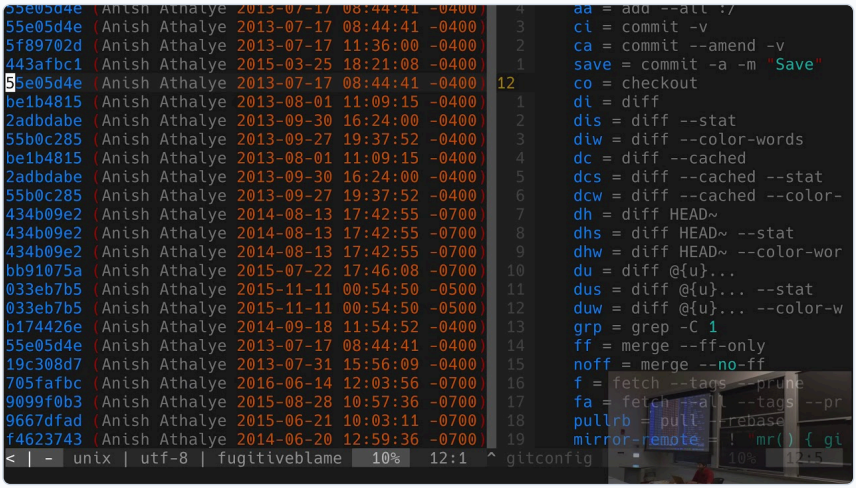
Resource / Description	URL	Micro QR
1. regex debugger	https://regex101.com/r/qqbZqh/2	
2. e-mail address	https://www.regular-expressions.info/email.html	
3. မလွယ်ကူပါ	https://web.archive.org/web/20221223174323/http://emailregex.com/	
4. ဆွေးနွေးငြင်းခုံမှုများစွာ	https://stackoverflow.com/questions/201323/how-to-validate-an-email-address-using-a-regular-expression/1917982	
5. Test များ ရေးသားခြင်း	https://fightingforalostcause.net/content/misc/2006/compare-email-regex.php	
6. Test matrices များ ပြုလုပ်ခြင်း	https://mathiasbynens.be/demo/url-regex	
7. သဘာဝကိန်း (Prime number) ဟုတ်မဟုတ်	https://www.noulakaz.net/2007/03/18/a-regular-expression-to-check-for-prime-numbers/	
8. အရာအားလုံးကို ပြုလုပ်နိုင်သောကြောင့်	https://web.archive.org/web/20251210045942/https://backreference.org/2010/02/10/idiomatic-awk/	
9. `st`	https://github.com/nferraz/st	
10. Plot တိုင်းတာပုံဆွဲခြင်း	https://ggplot2.tidyverse.org/	
11. ဤနေရာ	https://regexone.com/	
12. ဤအရာ	https://commons.wikimedia.org/wiki/Data:Wikipedia_statistics/data.tab	
13. ဤအရာ	https://ucr.fbi.gov/crime-in-the-u.s/2016/crime-in-the-u.s.-2016/topic-pages/tables/table-1	
14. ဤနေရာမှ	https://www.springboard.com/blog/data-science/free-public-data-sets-data-science-project/	
15. `pup`	https://github.com/EricChiang/pup	

Resource / Description	URL	Micro QR
16. `jq`	https://stedolan.github.io/jq/	

Dotfiles

▶ LECTURE VIDEO REFERENCE

Dotfiles



Scan the QR code below using your mobile camera or click the link to watch this full lecture online:

Direct Online Link:

https://www.youtube.com/watch?v=YSZBWWJw3mI

ပရိုဂရမ်အများအပြားကို “dotfiles” ဟုခေါ်သော plain-text ဖိုင်များအသုံးပြု၍ ချိန်ညှိပြင်ဆင်ကြပါတယ် (ဖိုင်အမည်များသည် `.` ဖြင့် စတင်လေ့ရှိသောကြောင့် ဖြစ်သည်၊ ဥပမာ `~/.gitconfig` ၊ သို့မှသာ ယင်းတို့သည် `ls` ဖြင့် directory အတွင်းရှိ ဖိုင်များကို ကြည့်ရှုသည့်အခါ ပုံသေအားဖြင့် ကွယ်နေမည်ဖြစ်သည်)။

သင်အသုံးပြုသည့် tool အတော်များများတွင် အသေးစိတ် ချိန်ညှိနိုင်သော စက်တင် (setting) အမြောက်အမြား ပါဝင်လေ့ရှိသည်။ ထို့အပြင် tool များကို သီးသန့် ဘာသာစကားများဖြင့် စိတ်ကြိုက်

ပြင်ဆင်လေ့ရှိကြသည်၊ ဥပမာ Vim အတွက် Vimscript သို့မဟုတ် shell အတွက် shell ၏ ကိုယ်ပိုင် ဘာသာစကား ဖြစ်သည်။

သင်၏ tool များကို သင်နှစ်သက်ရာ လုပ်ငန်းစဉ် (workflow) နှင့်အညီ စိတ်ကြိုက် ပြင်ဆင် (customize) ပြီး အံဝင်ခွင်ကျ ဖြစ်အောင် ညှိယူခြင်းက သင့်ကို ပိုမို အလုပ်တွင်ကျယ်စေပါလိမ့်မည်။ GitHub မှ အခြားသူတစ်ဦး၏ dotfile များကို ကူးယူ (clone) အသုံးပြုခြင်းထက် မိမိ၏ tool များကို မိမိကိုယ်တိုင် စိတ်ကြိုက် ပြင်ဆင်ရန် အချိန်ပေး ရင်းနှီးမြှုပ်နှံဖို့ ကျွန်ုပ်တို့ အကြံပြုလိုပါသည်။

သင့်ထံတွင် dotfile အချို့ တည်ဆောက်ထားပြီးဖြစ်နိုင်ပါသည်။ ရှာဖွေကြည့်ရှုနိုင်သည့် နေရာအချို့မှာ-

- `~/.bashrc`
- `~/.emacs`
- `~/.vim`
- `~/.gitconfig`

ပရိုဂရမ်အချို့သည် ဖိုင်များကို သင်၏ home folder အောက်တွင် တိုက်ရိုက်မထားဘဲ `~/.config` အောက်ရှိ folder တစ်ခုအတွင်း ထည့်သွင်းထားလေ့ရှိကြသည်။

Dotfile များသည် command line application များအတွက်သာ သီးသန့်မဟုတ်ပါ၊ ဥပမာအားဖြင့် **MPV** (https://mpv.io/) မီဒီယိုပလေယာကို `~/.config/mpv` အောက်ရှိ ဖိုင်များကို ပြင်ဆင်ခြင်းဖြင့် စက်တင်ညှိနိုင်ပါသည်။

Tool များကို စိတ်ကြိုက် ပြင်ဆင်ရန် လေ့လာခြင်း (Learning to customize tools)

အွန်လိုင်း documentation များ သို့မဟုတ် **man pages** (https://en.wikipedia.org/wiki/Man_page) များကို ဖတ်ရှုခြင်းဖြင့် သင်၏ tool ၏ စက်တင်များအကြောင်း လေ့လာနိုင်ပါသည်။ အခြား ကောင်းမွန်သော နည်းလမ်းတစ်ခုမှာ အတိအကျ ပရိုဂရမ်များအကြောင်း ရေးသားထားသည့် ဘလော့ဂ်ဆောင်းပါးများကို အင်တာနက်တွင် ရှာဖွေခြင်း ဖြစ်ပြီး ယင်းတို့၌ စာရေးသူများ၏ နှစ်သက်ရာ စိတ်ကြိုက်ပြင်ဆင်မှုများအကြောင်း ရေးသားထားကြသည်။ စိတ်ကြိုက်ပြင်ဆင်မှုများအကြောင်း လေ့လာရန် အခြား နည်းလမ်းတစ်ခုမှာ အခြားသူများ၏ dotfile များကို ဝင်ရောက်ကြည့်ရှုခြင်း ဖြစ်သည်- GitHub တွင် **dotfiles repositories** (https://github.com/search?o=desc&q=dotfiles&s=stars&type=Repositories) အမြောက်အမြားကို ရှာဖွေတွေ့ရှိနိုင်ပါသည် — လူကြိုက်အများဆုံး repository ကို **ဒီမှာ** (https://github.com/mathiasbynens/dotfiles) ကြည့်ရှုပါ (စက်တင်များကို မျက်စိမှိတ် အကန်းလိုက် မကူးယူရန်တော့ ကျွန်ုပ်တို့ အကြံပြုလိုပါသည်)။

စနစ်တကျ ဖွဲ့စည်း စီမံခြင်း (Organization)

သင်၏ dotfile များကို မည်သို့ စနစ်တကျ ဖွဲ့စည်းထားသင့်သနည်း။ ယင်းတို့ကို သီးသန့် folder တစ်ခုအတွင်း ထည့်သွင်းထားပြီး၊ version control အောက်တွင် ထားရှိကာ၊ script တစ်ခုအသုံးပြု၍ သက်ဆိုင်ရာ နေရာများသို့ **symlink** ချိတ်ဆက်ထားသင့်ပါသည်။ ဤသို့ ပြုလုပ်ခြင်း၏ အကျိုးကျေးဇူးများမှာ-

- **လွယ်ကူစွာ တပ်ဆင်နိုင်ခြင်း (Easy installation)**: စက်အသစ်တစ်လုံးသို့ ဝင်ရောက်အသုံးပြုသည့်အခါ သင်၏ စိတ်ကြိုက်ပြင်ဆင်မှုများကို အသုံးပြုနိုင်ရန် မိနစ်ပိုင်းမျှသာ ကြာမြင့်မည် ဖြစ်သည်
- **နေရာမရွေး အသုံးပြုနိုင်ခြင်း (Portability)**: သင်၏ tool များသည် ဘယ်နေရာမှာမဆို ပုံစံတူအတိုင်း အလုပ်လုပ်ဆောင်မည် ဖြစ်သည်
- **ဟန်ချက်ညီ ပေါင်းစပ်နိုင်ခြင်း (Synchronization)**: သင်၏ dotfile များကို မည်သည့်နေရာတွင်မဆို update ပြုလုပ်နိုင်ပြီး အားလုံးကို ဟန်ချက်ညီညီ ထိန်းသိမ်းထားနိုင်သည်
- **အပြောင်းအလဲများကို စောင့်ကြည့်မှတ်တမ်းတင်နိုင်ခြင်း (Change tracking)**: သင်၏ Programmer သက်တမ်းတစ်ခုလုံးတွင် dotfile များကို ထိန်းသိမ်းသွားရဖွယ် ရှိသဖြင့်၊ ကာလရှည် စီမံကိန်းများအတွက် version history ရှိထားခြင်းသည် အလွန်ကောင်းမွန်ပါသည်

```
cd ~/src
mkdir dotfiles
cd dotfiles
git init
touch bashrc
# create a bashrc with some settings, e.g.:
#     PS1='\w > '
touch install
chmod +x install
# insert the following into the install script:
#     #!/usr/bin/env bash
#     BASEDIR=$(dirname $0)
#     cd $BASEDIR
#
#     ln -s ${PWD}/bashrc ~/.bashrc
git add bashrc install
git commit -m 'Initial commit'
```

အဆင့်မြင့် ခေါင်းစဉ်များ (Advanced topics)

စက်အလိုက် သီးသန့် စိတ်ကြိုက်ပြင်ဆင်ခြင်းများ (Machine-specific customizations)

အများအားဖြင့် စက်အမျိုးမျိုးတွင် ပုံစံတူ configuration ကိုသာ အသုံးပြုလိုကြသော်လည်း၊ အချို့အချိန်များတွင် သီးခြား စက်တစ်ခုအတွက် အနည်းငယ် ကွဲပြားမှု (delta) ကို လိုချင်နိုင်ပါသည်။ ဤအခြေအနေကို ဖြေရှင်းနိုင်မည့် နည်းလမ်းအချို့မှာ အောက်ပါအတိုင်း ဖြစ်သည်-

စက်အလိုက် Branch တစ်ခုစီ ခွဲထားခြင်း (Branch per machine)

စက်တစ်လုံးစီအတွက် branch တစ်ခုစီ ထိန်းသိမ်းရန် version control ကို အသုံးပြုပါ။ ဤနည်းလမ်းသည် ယုတ္တိဗေဒအရ ရှင်းလင်းတိုက်ရိုက်သော်လည်း အနည်းငယ် ဝန်လေး (heavyweight) စေနိုင်ပါသည်။

If statement များ အသုံးပြုခြင်း (If statements)

Configuration ဖိုင်က ထောက်ပံ့ပေးပါက စက်အလိုက် သီးသန့် စိတ်ကြိုက်ပြင်ဆင်မှုများ ထည့်သွင်းရန် if-statement နှင့် ညီမျှသော နည်းလမ်းကို အသုံးပြုပါ။ ဥပမာအားဖြင့် သင်၏ shell တွင် အောက်ပါကဲ့သို့ ရေးသားနိုင်သည်-

```
if [[ "$(uname)" == "Linux" ]]; then {do_something else}; fi

# Darwin is the architecture name for macOS systems
if [[ "$(uname)" == "Darwin" ]]; then {do_something}; fi

# You can also make it machine specific
if [[ "$(hostname)" == "myServer" ]]; then {do_something}; fi
```

Include များကို အသုံးပြုခြင်း (Includes)

Configuration ဖိုင်က ထောက်ပံ့ပေးပါက include များကို အသုံးပြုပါ။ ဥပမာအားဖြင့် `~/.gitconfig` တွင် အောက်ပါ စက်တင် ထည့်သွင်းထားနိုင်သည်-

```
[include]
    path = ~/.gitconfig_local
```

ထို့နောက် စက်တစ်လုံးစီတွင် `~/.gitconfig_local` ၌ စက်အလိုက် သီးသန့် စက်တင်များ ပါဝင်နိုင်ပါသည်။ စက်အလိုက် သီးသန့် စက်တင်များအတွက် ၎င်းတို့ကို သီးသန့် repository တစ်ခုတွင်ပင် ထိန်းသိမ်းစောင့်ကြည့် (track) နိုင်ပါသည်။

အခြားသော ပရိုဂရမ်များတွင် စက်တင်အချို့ကို မျှဝေသုံးစွဲလိုပါကလည်း ဤနည်းလမ်းသည် အသုံးဝင်ပါသည်။ ဥပမာအားဖြင့် `bash` နှင့် `zsh` နှစ်ခုစလုံးကို တူညီသော alias အစုအဝေး မျှဝေသုံးစွဲစေလိုပါက ယင်းတို့ကို `.aliases` အောက်တွင် ရေးသားနိုင်ပြီး နှစ်ခုစလုံးတွင် အောက်ပါ block ကို ထည့်သွင်းထားနိုင်ပါသည်။

```
# Test if ~/.aliases exists and source it
if [ -f ~/.aliases ]; then
    source ~/.aliases
fi
```

လေ့လာရန် သယံဇာတများ (Resources)

- သင်၏ နည်းပြများ၏ dotfile များ- Anish (https://github.com/anishathalye/dotfiles), Jon (https://github.com/jonhoo/configs), Jose (https://github.com/jjgo/dotfiles)
- GitHub does dotfiles (https://dotfiles.github.io/)- dotfile framework များ၊ utility များ၊ ဥပမာများနှင့် သင်ခန်းစာများ
- Shell startup scripts (https://web.archive.org/web/20260329133158/https://blog.flowblok.id.au/2013-02/shell-startup-scripts.html)- သင်၏ shell အတွက် အသုံးပြုသည့် မတူညီသော configuration ဖိုင်များအကြောင်း ရှင်းလင်းချက်

လေ့ကျင့်ခန်းများ (Exercises)

1. သင်၏ dotfile များအတွက် folder တစ်ခု ဖန်တီးပြီး **version control** (https://missing-semester-my.github.io/2019/version-control/) ကို စတင် တည်ဆောက်ပါ။

2. အနည်းဆုံး ပရိုဂရမ် တစ်ခုအတွက် (ဥပမာ သင်၏ shell) စိတ်ကြိုက်ပြင်ဆင်မှု ပါဝင်သော configuration တစ်ခု ထည့်သွင်းပါ (စတင်သည့်အနေဖြင့် `$PS1` ကို သတ်မှတ်၍ သင်၏ shell prompt ကို စိတ်ကြိုက် ပြင်ဆင်ခြင်းကဲ့သို့ ရိုးရှင်းသော အရာတစ်ခု ဖြစ်နိုင်သည်)။

3. စက်အသစ်တစ်လုံးတွင် သင်၏ dotfile များကို လျင်မြန်စွာ (လူကိုယ်တိုင် အားစိုက်ထုတ်ရန် မလိုဘဲ) တပ်ဆင်နိုင်မည့် နည်းလမ်းတစ်ခု စီစဉ်ပါ။ ၎င်းသည် ဖိုင်တစ်ခုစီအတွက် `ln -s` ကို ခေါ်ယူပေးသည့် shell script တစ်ခုကဲ့သို့ ရိုးရှင်းနိုင်သည်၊ သို့မဟုတ် **သီးသန့် utility** (https://dotfiles.github.io/utilities/) တစ်ခုကိုလည်း အသုံးပြုနိုင်ပါသည်။

4. သင်၏ တပ်ဆင်မှု script ကို virtual machine အသစ်တစ်ခုတွင် စမ်းသပ်ပါ။

5. လက်ရှိ tool စက်တင်များ အားလုံးကို သင်၏ dotfiles repository သို့ ပြောင်းရွှေ့ပါ။

6. သင်၏ dotfile များကို GitHub ပေါ်တွင် လွှင့်တင်ပါ (publish ပြုလုပ်ပါ)။

🔗 External Links & References (ပြင်ပ လင့်ခ်များနှင့် ကိုးကားချက်များ)

Resource / Description	URL	Micro QR
1. MPV	https://mpv.io/	
2. man pages	https://en.wikipedia.org/wiki/Man_page	
3. dotfiles repositories	https://github.com/search?o=desc&q=dotfiles&s=stars&type=Repositories	
4. ဒီမှာ	https://github.com/mathiasbynens/dotfiles	
5. Anish	https://github.com/anishathalye/dotfiles	
6. Jon	https://github.com/jonhoo/configs	
7. Jose	https://github.com/jjgo/dotfiles	
8. GitHub does dotfiles	https://dotfiles.github.io/	
9. Shell startup scripts	https://web.archive.org/web/20260329133158/https://blog.flowblok.id.au/2013-02/shell-startup-scripts.html	
10. version control	https://missing-semester-my.github.io/2019/version-control/	
11. သီးသန့် utility	https://dotfiles.github.io/utilities/	

Editors

▶ LECTURE VIDEO REFERENCE

Editors

```
layout: page
title: "Editors"
presenter: Anish
---

# Importance of Editors

As programmers, we spend most of our time editing plain-text files. It's worth
investing time learning an editor that fits your needs.

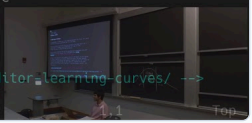
How do you learn a new editor? You force yourself to use that editor for a
while, even if it hampers your productivity. It'll pay off soon
enough (two weeks is enough to learn the basics).

We are going to teach you Vim, but we encourage you to experiment with other
editors. It's a very personal choice, and people have [strong
opinions](https://en.wikipedia.org/wiki/Editor_war).

We can't teach you how to use a powerful editor in 50 minutes, so we're going
to focus on teaching you the basics, showing you some of the more advanced
functionality, and giving you the resources to master the tool. We'll teach you
lessons in the context of Vim, but most ideas will translate to any other
powerful editor you use (and if they don't, then you probably shouldn't use
that editor!).

![Editor Learning Curves](/static/media/editor-learning-curves.jpg)

<!-- source: https://blogs.msdn.microsoft.com/steverowe/2004/11/17/code-editor-learning-curves/ -->
```



Scan the QR code below using your mobile camera or click the link to watch this full lecture online:

Direct Online Link:

https://www.youtube.com/watch?v=1vLcusYSrI4

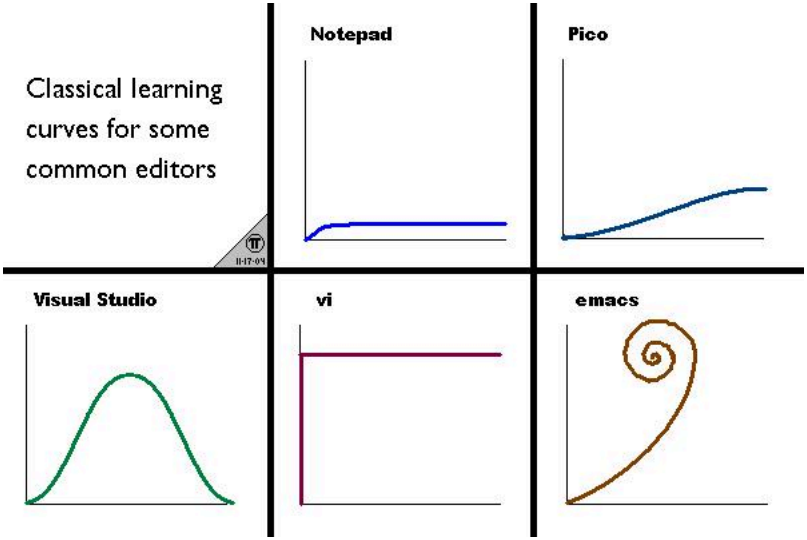
စာတည်းဆော့ဖ်ဝဲများ (Editors) ၏ အရေးပါမှု

Programmer များအနေဖြင့် ကျွန်ုပ်တို့သည် ကျွန်ုပ်တို့၏ အချိန်အများစုကို Plain-text ဖိုင်များကို ပြင်ဆင်ရေးသားခြင်း (editing) ဖြင့် ကုန်လွန်စေကြသည်။ မိမိ၏ လိုအပ်ချက်နှင့် ကိုက်ညီသော Editor တစ်ခုကို လေ့လာရန် အချိန်ပေး ရင်းနှီးမြှုပ်နှံရကျိုး နပ်ပါသည်။

Editor အသစ်တစ်ခုကို မည်သို့ လေ့လာမည်နည်း။ ထို Editor ကို ခဏတာ အသုံးပြုရုန် မိမိကိုယ်ကို တိုက်တွန်းအားထုတ်ရမည်ဖြစ်ပြီး၊ ၎င်းသည် သင်၏ လုပ်ဆောင်နိုင်စွမ်း (productivity) ကို ခေတ္တခဏ နှောင့်နှေးစေလျှင်ပင် အသုံးပြုရမည်ဖြစ်သည်။ မကြာမီ အကျိုးကျေးဇူး ပြန်လည် ရရှိလာမည်ဖြစ်သည် (အခြေခံများကို လေ့လာရန် သီတင်းပတ်နှစ်ပတ်ခန့်မျှ ပေးပါက လုံလောက်ပါသည်)။

ကျွန်ုပ်တို့သည် သင်တို့အား Vim အကြောင်း သင်ကြားပေးမည်ဖြစ်သော်လည်း အခြားသော Editor များကိုလည်း စမ်းသပ်ကြည့်ရှုရန် တိုက်တွန်းပါသည်။ ၎င်းသည် တစ်ဦးချင်းစီ၏ ကြိုက်နှစ်သက်မှုအပေါ် များစွာမူတည်ပြီး လူများသည် [သဘောထား ကွဲလွဲမှုများစွာ](https://en.wikipedia.org/wiki/Editor_war) ရှိကြသည်။

မိနစ် ၅၀ အတွင်း စွမ်းအားထက်မြက်သော Editor တစ်ခု အသုံးပြုနည်းကို သင်ကြားမပေးနိုင်သောကြောင့် ကျွန်ုပ်တို့သည် အခြေခံများကို သင်ကြားပေးခြင်း၊ ပို့မို့ အဆင့်မြင့်သော လုပ်ဆောင်ချက်အချို့ကို ပြသပေးခြင်းနှင့် ဤ tool ကို ကျွမ်းကျင်စွာ အသုံးပြုနိုင်ရန် အရင်းအမြစ်များ မျှဝေပေးခြင်းတို့အပေါ် အဓိကထားသွားမည်ဖြစ်သည်။ ကျွန်ုပ်တို့သည် Vim ကို နမူနာထား၍ သင်ခန်းစာများကို သင်ကြားပေးမည်ဖြစ်သော်လည်း အယူအဆ အများစုသည် သင်အသုံးပြုသည့် အခြားသော စွမ်းအားထက်မြက်သည့် Editor များသို့လည်း ပေါင်းစပ်အသုံးပြုနိုင်မည်ဖြစ်သည် (ထိုသို့ ပေါင်းစပ်အသုံးပြု၍ မရပါက ထို Editor ကို သင် အသုံးမပြုသင့်ပါ!)။



Editor Learning Curves

Editor သင်ယူမှုဆိုင်ရာ ဂရပ် (learning curves graph) ဆိုသည်မှာ ဒဏ္ဍာရီ (myth) တစ်ခုသာ ဖြစ်သည်။ စွမ်းအားထက်မြက်သော Editor တစ်ခု၏ အခြေခံများကို လေ့လာခြင်းသည် အလွန်ပင် လွယ်ကူပါသည် (ကျွမ်းကျင်ပိုင်နိုင်ရန် နှစ်ပေါင်းများစွာ ကြာမြင့်နိုင်သော်လည်း)။

ယနေ့ခေတ်တွင် မည်သည့် Editor များ ရေပန်းစားသနည်း။ ဤ [Stack Overflow စမ်းသပ်စစ်တမ်း](https://insights.stackoverflow.com/survey/2018/#development-environments-and-tools) ကို ကြည့်ပါ (Stack Overflow အသုံးပြုသူများသည် Programmer အားလုံးကို ကိုယ်စားမပြုနိုင်သောကြောင့် အနည်းငယ် ဘက်လိုက်မှု ရှိနိုင်သည်)။

Command-line Editors များ

နောက်ဆုံးတွင် GUI editor တစ်ခုကို အသုံးပြုရန် ဆုံးဖြတ်လိုက်လျှင်ပင် Remote machine များပေါ်ရှိ ဖိုင်များကို လွယ်ကူစွာ ပြင်ဆင်နိုင်ရန် Command-line editor တစ်ခုကို လေ့လာထားရကျိုး နပ်ပါသည်။

Nano

Nano သည် ရိုးရှင်းသော Command-line editor တစ်ခု ဖြစ်သည်။

- Arrow key များဖြင့် ရွှေ့လျားနိုင်သည်
- အခြားသော Shortcut များ (save, exit) အားလုံးကို အောက်ခြေတွင် ပြသထားသည်

Vim

Vi/Vim သည် စွမ်းအားထက်မြက်သော Text editor တစ်ခု ဖြစ်သည်။ ၎င်းသည် နေရာအများစုတွင် မူလ အတိုင်း ထည့်သွင်းလေ့ရှိသည့် Command-line ပရိုဂရမ်တစ်ခု ဖြစ်သောကြောင့် Remote machine ပေါ်ရှိ ဖိုင်များကို ပြင်ဆင်ရန် အလွန် အဆင်ပြေစေသည်။

Vim တွင် GVim နှင့် [MacVim](https://macvim-dev.github.io/macvim/) ကဲ့သို့သော Graphical version များလည်း ရှိသည်။ ၎င်းတို့သည် 24-bit color၊ Menu များ၊ Popup များကဲ့သို့သော အပိုဆောင်း Feature များကို ထောက်ပံ့ပေးသည်။

Vim ၏ သဘောတရား (Philosophy of Vim)

- ပရိုဂရမ်မင်း ရေးသားသည့်အခါ သင်၏ အချိန်အများစုကို စာရေးသားခြင်းထက် စာဖတ်ခြင်း/ပြင်ဆင်ခြင်း (reading/editing) ဖြင့် ကုန်လွန်စေသည်

```
- Vim သည် **modal** editor တစ်ခု ဖြစ်သည်- စာသားများ ထည့်သွင်းခြင်း (inserting text) နှင့် စာသားများကို ပြုပြင်ခြင်း (manipulating text) တို့အတွက် သီးခြား Mode များ ရှိသည်
```

- Vim ကို ပရိုဂရမ် ရေးသား၍ စီမံနိုင်သည် (Vimscript အပြင် Python ကဲ့သို့သော အခြား Language များ နှင့်လည်း ပြုလုပ်နိုင်သည်)
- Vim ၏ Interface ကိုယ်တိုင်က Programming language တစ်ခုကဲ့သို့ ဖြစ်သည်

```
- Keystroke များ (မှတ်မိလွယ်သော Mnemonic အမည်များဖြင့်) သည် Command များ ဖြစ်ကြသည်
- Command များကို ပေါင်းစပ်အသုံးပြုနိုင်သည် (composable)
```

- Mouse ကို မသုံးပါနှင့်- အလွန် နှေးကွေးသည်
- Editor သည် သင် စဉ်းစားသည့် အရှိန်အတိုင်း လုပ်ဆောင်နိုင်ရမည်

Vim အခြေခံ မိတ်ဆက် (Introductory Vim)

Mode များ

Vim သည် လက်ရှိ Mode ကို ဘယ်ဘက် အောက်ခြေတွင် ပြသပေးသည်။

- Normal mode- ဖိုင်အတွင်း ရွှေ့လျားရန်နှင့် ပြင်ဆင်မှုများ ပြုလုပ်ရန်

```
- သင်၏ အချိန်အများစုကို ဤနေရာတွင် ကုန်လွန်မည် ဖြစ်သည်
```

- Insert mode- စာသားများ ထည့်သွင်းရန်
- Visual (visual, line, သို့မဟုတ် block) mode- စာသား Block များကို ရွေးချယ် (select) ရန်

မည်သည့် Mode မှမဆို Normal mode သို့ ပြန်လည် ပြောင်းလဲရန် `<ESC>` ကို နှိပ်၍ Mode ပြောင်းနိုင်သည်။ Normal mode မှ Insert mode သို့ `i` ဖြင့်လည်းကောင်း၊ Visual mode သို့ `v` ဖြင့်လည်းကောင်း၊ Visual line mode သို့ `V` ဖြင့်လည်းကောင်း၊ Visual block mode သို့ `<C-v>` ဖြင့်လည်းကောင်း ဝင်ရောက်နိုင်သည်။

Vim ကို အသုံးပြုသည့်အခါ `<ESC>` key ကို အများအပြား အသုံးပြုရသဖြင့် Caps Lock key ကို Escape သို့ Remap ပြုလုပ်ရန် စဉ်းစားပါ။

အခြေခံများ

Vim ex command များကို Normal mode တွင် `:{command}` ဖြင့် ရိုက်နှိပ် ထုတ်ပြန်သည်။

- `:q` ထွက်မည် (Window ပိတ်မည်)
- `:w` သိမ်းဆည်းမည် (Save)
- `:wq` သိမ်းဆည်းပြီး ထွက်မည်
- `:e {name of file}` ပြင်ဆင်ရန် ဖိုင်ကို ဖွင့်မည်
- `:ls` ဖွင့်ထားသော Buffer များကို ပြသမည်
- `:help {topic}` အကူအညီ ဖွင့်မည်

```
- `:help :w` သည် `:w` ex command အတွက် Help ကို ဖွင့်ပေးသည်
- `:help w` သည် `w` ရွှေ့လျားမှု (movement) အတွက် Help ကို ဖွင့်ပေးသည်
```

ရွှေ့လျားခြင်း (Movement)

Vim သည် ထိရောက်သော ရွှေ့လျားမှု (efficient movement) ဖြင့် အဓိက လည်ပတ်သည်။ Normal mode တွင် ဖိုင်အတွင်း လမ်းကြောင်းရှာ သွားလာပါ။

- အကျင့်ဆိုးများကို ရှောင်ရှားရန် Arrow key များကို ပိတ်ထားပါ

```
nnoremap <Left> :echoe "Use h"<CR>
nnoremap <Right> :echoe "Use l"<CR>
nnoremap <Up> :echoe "Use k"<CR>
nnoremap <Down> :echoe "Use j"<CR>
```

- အခြေခံ ရွှေ့လျားမှု- `hjkl` (ဘယ်၊ အောက်၊ အထက်၊ ညာ)
- စကားလုံးများ- `w` (နောက်စကားလုံး)၊ `b` (စကားလုံး၏ အစ)၊ `e` (စကားလုံး၏ အဆုံး)
- စာကြောင်းများ- `0` (စာကြောင်း၏ အစ)၊ `^` (ပထမဆုံး ကွက်လပ်မဟုတ်သော စာလုံး)၊ `$` (စာကြောင်း၏ အဆုံး)
- မျက်နှာပြင်- `H` (မျက်နှာပြင်၏ အထိပ်)၊ `M` (မျက်နှာပြင်၏ အလယ်)၊ `L` (မျက်နှာပြင်၏ အောက်ခြေ)
- ဖိုင်- `gg` (ဖိုင်၏ အစ)၊ `G` (ဖိုင်၏ အဆုံး)
- စာကြောင်း နံပါတ်များ- `:{number}<CR>` သို့မဟုတ် `{number}G` (စာကြောင်း နံပါတ် {number})
- အထွေထွေ- `%` (ကိုက်ညီသော အရာ/တွဲဖက် ကိုက်ညီမှု)
- ရှာဖွေမှု- `f{character}`, `t{character}`, `F{character}`, `T{character}`

```
- လက်ရှိ စာကြောင်းပေါ်တွင် ရှေ့သို့/နောက်သို့ {character} ကို ရှာမည်/အရောက်သွားမည်
```

- N ကြိမ် ထပ်မံပြုလုပ်ခြင်း- `{number}{movement}` ။ ဥပမာ `10j` သည် အောက်သို့ စာကြောင်း ၁၀ ကြောင်း ရွှေ့သည်
- ရှာဖွေခြင်း- `/{regex}`, ကိုက်ညီမှုများကို ရွှေ့လျားကြည့်ရှုရန် `n` / `N`

ရွေးချယ်ခြင်း (Selection)

Visual mode များ-

- Visual
- Visual Line

- Visual Block

ရွေးချယ်မှုပြုလုပ်ရန် ရွှေ့လျားမှု Key (Movement keys) များကို အသုံးပြုနိုင်သည်။

စာသားများကို ပြုပြင်ခြင်း (Manipulating text)

ယခင်က Mouse ဖြင့် ပြုလုပ်ခဲ့သမျှ အရာအားလုံးကို ယခုအခါ Keyboard များ (နှင့် စွမ်းအားထက်မြက်သည့် ပေါင်းစပ်အသုံးပြုနိုင်သော Command များ) ဖြင့် ပြုလုပ်မည်ဖြစ်သည်။

- `i` Insert mode သို့ ဝင်မည်

```
- သို့သော် စာသားများကို ပြုပြင်ရန်/ဖျက်ပစ်ရန်အတွက် Backspace ထက် ပိုမိုကောင်းမွန်သည့် အရာတစ်ခုခု ကို အသုံးပြုရန် လိုအပ်သည်
```

- `o` / `O` အောက်ခြေ / အထက်တွင် စာကြောင်းအသစ် ထည့်မည်
- `d{motion}` {motion} အတိုင်း ဖျက်မည်

```
- ဥပမာ `dw` သည် စကားလုံးကို ဖျက်မည်၊ `d$` သည် စာကြောင်းအဆုံးထိ ဖျက်မည်၊ `d0` သည် စာကြောင်းအစထိ ဖျက်မည်
```

- `c{motion}` {motion} အတိုင်း ပြောင်းလဲမည်

```
- ဥပမာ `cw` သည် စကားလုံးကို ပြောင်းလဲမည်
- `d{motion}` ၏ နောက်တွင် `i` ရိုက်နှိပ်လိုက်သကဲ့သို့ ဖြစ်သည်
```

- `x` စာလုံး (character) ကို ဖျက်မည် (`dl` နှင့် ညီမျှသည်)
- `s` စာလုံး (character) ကို အစားထိုးမည် (`xi` နှင့် ညီမျှသည်)
- Visual mode + ပြုပြင်ခြင်း

```
- စာသားကို ရွေးချယ်ပြီး ဖျက်ရန် `d` သို့မဟုတ် ပြောင်းလဲရန် `c` ကို နှိပ်ပါ
```

- Undo ပြုလုပ်ရန် `u` ၊ Redo ပြုလုပ်ရန် `<C-r>`
- လေ့လာရန် အခြားအရာများစွာ ရှိပါသေးသည်- ဥပမာ `~` သည် စာလုံး၏ Case (စာလုံးကြီး/စာလုံးသေး) ကို ပြောင်းလဲပေးသည်

အရင်းအမြစ်များ (Resources)

- `vimtutor` Vim ကို သင်ကြားပေးသည့် Command-line ပရိုဂရမ်
- Vim ကို လေ့လာရန် **Vim Adventures** (https://vim-adventures.com/) ဂိမ်း

Vim ကို စိတ်ကြိုက် ပြင်ဆင်ခြင်း (Customizing Vim)

Vim ကို `~/.vimrc` ရှိ Plain-text Configuration ဖိုင် (Vimscript command များ ပါဝင်သည်) ဖြင့် စိတ်ကြိုက် ပြင်ဆင်နိုင်သည်။ သင် ဖွင့်ထားလိုသည့် အခြေခံ Setting အများအပြား ရှိနိုင်ပါသည်။

စိတ်ကူးစိတ်သန်းများ ရရှိရန် GitHub ရှိ အခြားသူများ၏ Dotfile များကို လေ့လာပါ၊ သို့သော် အခြားသူများ၏ Configuration တစ်ခုလုံးကို Copy-and-paste မလုပ်မိပါစေနှင့်။ ၎င်းကို ဖတ်ပါ၊ နားလည်အောင် လုပ်ပါ၊ ထို့နောက် သင် လိုအပ်သည်များကိုသာ ယူပါ။

စဉ်းစားသင့်သည့် စိတ်ကြိုက်ပြင်ဆင်မှု (Customization) အချို့-

- Syntax highlighting- `syntax on`
- Color scheme များ
- စာကြောင်း နံပါတ်များ- `set nu` / `set rnu`
- အရာအားလုံးကို Backspace ဖြင့် ဖျက်နိုင်ခြင်း- `set backspace=indent,eol,start`

အဆင့်မြင့် Vim (Advanced Vim)

ဤနေရာတွင် Editor ၏ စွမ်းအားကို ပြသရန် နမူနာ အနည်းငယ် ဖော်ပြထားပါသည်။ ဤသို့သော အရာအားလုံးကို ကျွန်ုပ်တို့ သင်ကြားမပေးနိုင်သော်လည်း သင် အသုံးပြုရင်းဖြင့် သင်ယူသွားရမည် ဖြစ်သည်။ ကောင်းမွန်သော နည်းလမ်းတစ်ခုမှာ- သင်၏ Editor ကို အသုံးပြုနေစဉ် "ဤအရာကို ပြုလုပ်ရန် ပိုမို ကောင်းမွန်သည့် နည်းလမ်း ရှိရမည်" ဟု စဉ်းစားမိပါက အမှန်တကယ် ရှိနေတတ်ပါသည်- အွန်လိုင်းတွင် ရှာဖွေကြည့်ပါ။

ရှာဖွေခြင်းနှင့် အစားထိုးခြင်း (Search and replace)

`:s` (substitute) command (**documentation** (https://vim.fandom.com/wiki/Search_and_replace))။

- `%s/foo/bar/g`

```
- ဖိုင်တစ်ပြင်လုံးတွင် foo ကို bar ဖြင့် အစားထိုးမည်
```

- `%s/\[.*\](\(.*\))/\1/g`

```
- အမည်တပ်ထားသော Markdown link များကို ရိုးရိုး URL များဖြင့် အစားထိုးမည်
```

Window အများအပြား အသုံးပြုခြင်း (Multiple windows)

- Window များကို ခွဲခြားရန် `sp` / `vsp`
- အတူတူပင်ဖြစ်သော Buffer ၏ View အများအပြားကို ကြည့်ရှုနိုင်သည်

Mouse ထောက်ပံ့မှု (Mouse support)

- `set mouse+=a`

```
- Click နှိပ်ခြင်း၊ Scroll လုပ်ခြင်း၊ ရွေးချယ်ခြင်းတို့ကို ပြုလုပ်နိုင်သည်
```

Macro များ

- Register `{character}` တွင် Macro စတင် အသံသွင်းရန် (record) `q{character}`
- Record လုပ်ခြင်း ရပ်တန့်ရန် `q`
- `@{character}` သည် Macro ကို ပြန်လည် မောင်းနှင်ပေးသည်
- Error ဖြစ်ပေါ်ပါက Macro လုပ်ဆောင်မှု ရပ်တန့်သွားမည်
- `{number}@{character}` သည် Macro ကို {number} ကြိမ် မောင်းနှင်ပေးသည်
- Macro များသည် Recursive (မိမိကိုယ်ကို ပြန်လည် ခေါ်ယူခြင်း) ဖြစ်နိုင်သည်

```
- ပထမဦးစွာ `q{character}q` ဖြင့် Macro ကို ရှင်းထုတ်ပါ
- Record လုပ်ခြင်း ပြီးစီးသည်အထိ (no-op ဖြစ်နေမည်ဖြစ်ပြီး) Macro ကို မောင်းနှင်ရန် `@{characte
r}` အသုံးပြု၍ အသံသွင်းပါ
```

- ဥပမာ- XML ကို JSON သို့ ပြောင်းလဲခြင်း (ဖိုင်)

```
- Key "name" / "email" ပါရှိသော Array of objects
- Python ပရိုဂရမ်တစ်ခု အသုံးပြုမလား။
- sed / regexes အသုံးပြုမလား
    - `g/people/d`
    - `%s/<person>/{/g`
    - `%s/<name>\(.*\)<\/name>/"name": "\1",/g`
    - ...
- Vim command များ / macro များ
    - ပထမဆုံးနှင့် နောက်ဆုံး စာကြောင်းများကို ဖျက်ရန် `Gdd`, `ggdd`
    - Element တစ်ခုတည်းကို Format ပြုလုပ်ရန် Macro (register `e`)
        - `<name>` ပါသော စာကြောင်းသို့ သွားပါ
        - `qe^r"f>s": "<ESC>f<C"<ESC>q`
    - Person တစ်ဦးကို Format ပြုလုပ်ရန် Macro
        - `<person>` ပါသော စာကြောင်းသို့ သွားပါ
        - `qpS{<ESC>j@eA,<ESC>j@ejS},<ESC>q`
    - Person တစ်ဦးကို Format ပြုလုပ်ပြီး နောက်တစ်ဦးထံ သွားရန် Macro
        - `<person>` ပါသော စာကြောင်းသို့ သွားပါ
        - `qq@pjq`
    - ဖိုင်အဆုံးထိ Macro ကို မောင်းနှင်ပါ
        - `999@q`
    - နောက်ဆုံး `,` ကို ကိုယ်တိုင် ဖျက်ထုတ်ပြီး `[` နှင့် `]` Delimiter များကို ထည့်သွင်းပါ
```

Vim ကို ပိုမို တိုးချဲ့ အသုံးပြုခြင်း (Extending Vim)

Vim ကို တိုးချဲ့ အသုံးပြုရန် Plugin များစွာ ရှိပါသည်။

ပထမဦးစွာ **vim-plug** (https://github.com/junegunn/vim-plug)၊ **Vundle** (https://github.com/VundleVim/Vundle.vim)၊ သို့မဟုတ် **pathogen.vim** (https://github.com/tpope/vim-pathogen) ကဲ့သို့သော Plugin manager တစ်ခုဖြင့် စတင် တပ်ဆင်ပါ။

စဉ်းစားသင့်သည့် Plugin အချို့-

- **ctrlp.vim** (https://github.com/kien/ctrlp.vim): fuzzy file finder
- **vim-fugitive** (https://github.com/tpope/vim-fugitive): git ပေါင်းစပ်အသုံးပြုခြင်း
- **vim-surround** (https://github.com/tpope/vim-surround): “surroundings” များကို ပြုပြင်ခြင်း
- **gundo.vim** (https://github.com/sjl/gundo.vim): undo tree တွင် လမ်းကြောင်းရှာခြင်း
- **nerdtree** (https://github.com/scrooloose/nerdtree): file explorer
- **syntastic** (https://github.com/vim-syntastic/syntastic): syntax စစ်ဆေးခြင်း

- **vim-easymotion** (https://github.com/easymotion/vim-easymotion): magic motion များ
- **vim-over** (https://github.com/osyo-manga/vim-over): substitute preview

Plugin စာရင်းများ-

- **Vim Awesome** (https://vimawesome.com/)

အခြားသော ပရိုဂရမ်များတွင် Vim-mode အသုံးပြုခြင်း

ရေပန်းစားသော Editor အများအပြားတွင် (ဥပမာ vim နှင့် emacs) အခြားသော Tool များစွာက Editor emulation ကို ထောက်ပံ့ပေးသည်။

- Shell

```
- bash: `set -o vi`
- zsh: `bindkey -v`
- `export EDITOR=vim` (`git` ကဲ့သို့သော ပရိုဂရမ်များ အသုံးပြုသည့် Environment variable)
```

- `~/.inputrc`

```
- `set editing-mode vi`
```

Web **browser များ** (https://vim.fandom.com/wiki/Vim_key_bindings_for_web_browsers) အတွက် Vim keybinding extension များပင် ရှိပါသည်၊ ရေပန်းစားသော Extension အချို့မှာ Google Chrome အတွက် **Vimium** (https://chrome.google.com/webstore/detail/vimium/dbepggeogbaibhgnhhndojpepiihcmeb?hl=en) နှင့် Firefox အတွက် **Tridactyl** (https://github.com/tridactyl/tridactyl) တို့ ဖြစ်ကြသည်။

အရင်းအမြစ်များ (Resources)

- **Vim Tips Wiki** (https://vim.fandom.com/wiki/Vim_Tips_Wiki)
- **Vim Advent Calendar** (https://vimways.org/2018/): အမျိုးမျိုးသော Vim အကြံပြုချက်များ
- **Neovim** (https://neovim.io/) သည် ပိုမို တက်ကြွစွာ တိုးတက်လျက်ရှိသော ခေတ်မီ Vim reimplementation ဖြစ်သည်။
- **Vim Golf** (https://www.vimgolf.com/): အမျိုးမျိုးသော Vim စိန်ခေါ်မှုများ

လေ့ကျင့်ခန်းများ

1. Editor အချို့ကို စမ်းသပ်ကြည့်ပါ။ အနည်းဆုံး Command-line editor တစ်ခု (ဥပမာ Vim) နှင့် အနည်းဆုံး GUI editor တစ်ခု (ဥပမာ Atom) ကို စမ်းသပ်ပါ။ `vimtutor` ကဲ့သို့သော သင်ခန်းစာများမှတစ်ဆင့် လေ့လာပါ (သို့မဟုတ် အခြား Editor များအတွက် သက်ဆိုင်ရာ သင်ခန်းစာများ)။ Editor အသစ်တစ်ခု၏ အမှန်တကယ် ခံစားချက်ကို ရရှိစေရန် သင်၏ အလုပ်များကို လုပ်ဆောင်နေစဉ် နှစ်ရက်ခန့် တောက်လျှောက် အသုံးပြုရန် သန္နိဋ္ဌာန်ချပါ။

2. သင်၏ Editor ကို စိတ်ကြိုက် ပြင်ဆင်ပါ။ အွန်လိုင်းရှိ အကြံပြုချက်များနှင့် နည်းလမ်းများကို ကြည့်ရှုပြီး အခြားသူများ၏ Configuration များကိုလည်း လေ့လာပါ (လေ့လာလွယ်အောင် ရေးသားထားလေ့ရှိသည်)။

3. သင်၏ Editor အတွက် Plugin များကို စမ်းသပ်ကြည့်ပါ။

4. အနည်းဆုံး သီတင်းပတ်အနည်းငယ်မျှ စွမ်းအားထက်မြက်သော Editor တစ်ခုကို သီးသန့် အသုံးပြုရန် သန္နိဋ္ဌာန်ချပါ- ထိုအချိန်တွင် အကျိုးကျေးဇူးများကို စတင် မြင်တွေ့ရမည်ဖြစ်သည်။ တစ်ချိန်ချိန်တွင် သင်၏ Editor သည် သင် စဉ်းစားသည့် အရှိန်အဟိုင်း အလုပ်လုပ်နိုင်မည် ဖြစ်သည်။

5. Linter တစ်ခု (ဥပမာ python အတွက် pyflakes) ကို ထည့်သွင်းပါ၊ ၎င်းကို သင်၏ Editor နှင့် ချိတ်ဆက်ပြီး အလုပ်လုပ်ပုံကို စမ်းသပ်ပါ။

🔗 External Links & References (ပြင်ပ လင့်ခ်များနှင့် ကိုးကားချက်များ)

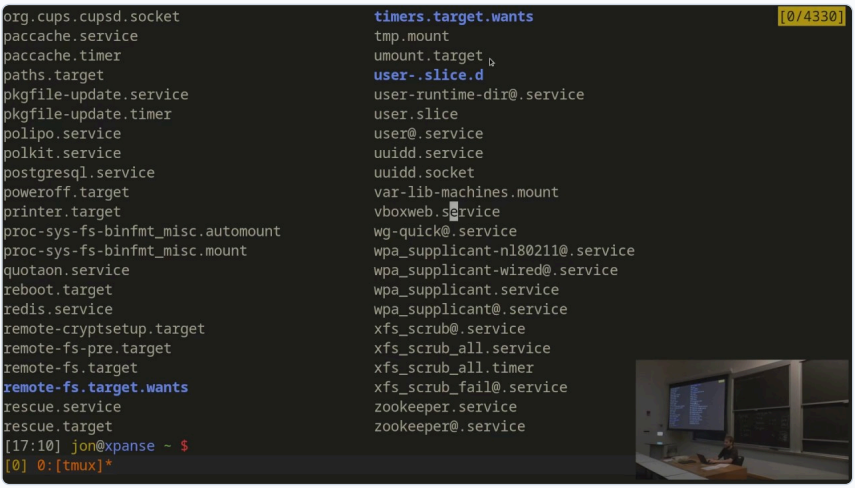
Resource / Description	URL	Micro QR
1. သဘောထား ကွဲလွဲမှုများစွာ	https://en.wikipedia.org/wiki/Editor_war	
2. Stack Overflow စမ်းသပ်စစ်တမ်း	https://insights.stackoverflow.com/survey/2018/#development-environments-and-tools	
3. MacVim	https://macvim-dev.github.io/macvim/	
4. Vim Adventures	https://vim-adventures.com/	
5. documentation	https://vim.fandom.com/wiki/Search_and_replace	
6. vim-plug	https://github.com/junegunn/vim-plug	
7. Vundle	https://github.com/VundleVim/Vundle.vim	
8. pathogen.vim	https://github.com/tpope/vim-pathogen	
9. ctrlp.vim	https://github.com/kien/ctrlp.vim	
10. vim-fugitive	https://github.com/tpope/vim-fugitive	
11. vim-surround	https://github.com/tpope/vim-surround	
12. gundo.vim	https://github.com/sjl/gundo.vim	
13. nerdtree	https://github.com/scrooloose/nerdtree	
14. syntastic	https://github.com/vim-syntastic/syntastic	
15. vim-easymotion	https://github.com/easymotion/vim-easymotion	

Resource / Description	URL	Micro QR
16. vim-over	https://github.com/osyo-manga/vim-over	
17. Vim Awesome	https://vimawesome.com/	
18. browser ပုဒ္	https://vim.fandom.com/wiki/Vim_key_bindings_for_web_browsers	
19. Vimium	https://chrome.google.com/webstore/detail/vimium/dbepggeogbaibhgnhhndojpepiihcmeb?hl=en	
20. Tridactyl	https://github.com/tridactyl/tridactyl	
21. Vim Tips Wiki	https://vim.fandom.com/wiki/Vim_Tips_Wiki	
22. Vim Advent Calendar	https://vimways.org/2018/	
23. Neovim	https://neovim.io/	
24. Vim Golf	https://www.vimgolf.com/	

Machine Introspection

▶ LECTURE VIDEO REFERENCE

Machine Introspection



Scan the QR code below using your mobile camera or click the link to watch this full lecture online:

Direct Online Link:

https://www.youtube.com/watch?v=eNYT2Oq3PF8

တစ်ခါတစ်ရံတွင် ကွန်ပျူတာများသည် ပုံမှန်မဟုတ်ဘဲ အမှားအယွင်းများ ဖြစ်ပေါ်တတ်သည်။ ထိုအခါ ဘာကြောင့် ထိုသို့ဖြစ်ရသည်ကို သင်သိရှိလိုပေမည်။ ထိုသို့ စစ်ဆေးရာတွင် ကူညီပေးနိုင်သည့် တူးလ် (tool) အချို့ကို လေ့လာကြည့်ကြပါစို့။

ဒါပေမဲ့ ပထမဦးစွာ သင်သည် စက်၏ အတွင်းပိုင်း စစ်ဆေးမှု (introspection) ကို ပြုလုပ်နိုင်စွမ်း ရှိမရှိ သေချာအောင် လုပ်ဆောင်ရန် လိုအပ်ပါသည်။ အများအားဖြင့် စနစ်၏ အတွင်းပိုင်းကို စစ်ဆေးရန်အတွက်

စနစ် ပိတ်ခြင်း (shutdown) အတွက် `power` group အဖွဲ့ဝင် ဖြစ်ရခြင်းကဲ့သို့သော သီးခြား လုပ်ပိုင်ခွင့် (privileges) များ ရှိရန် လိုအပ်ပါသည်။ `root` အသုံးပြုသူ (user) သည် အမြင့်ဆုံး လုပ်ပိုင်ခွင့် ရှိသူဖြစ်ပြီး မည်သည့်အရာကိုမဆို နီးပါး လုပ်ဆောင်နိုင်ပါသည်။ သင်သည် command တစ်ခုကို `sudo` အသုံးပြု၍ `root` အဖြစ် သွားရောက် လုပ်ဆောင်နိုင်ပါသည် (သို့သော် သတိထားရန် လိုအပ်ပါသည်။)

ဘာတွေ ဖြစ်ပျက်ခဲ့သလဲ။

တစ်ခုခု အမှားအယွင်း ဖြစ်သွားပါက ပထမဦးစွာ စတင်ကြည့်ရှုရမည့် နေရာမှာ အမှား ဖြစ်ပွားခဲ့သည့် အချိန် ဝန်းကျင်တွင် ဘာတွေ ဖြစ်ပျက်ခဲ့သလဲ ဆိုသည်ပင် ဖြစ်သည်။ ၎င်းအတွက် ကျွန်ုပ်တို့သည် လော့ဂ် (log) များကို ကြည့်ရှုရန် လိုအပ်ပါသည်။

ရှေးယခင်က လော့ဂ် (log) များကို `/var/log` ထဲတွင် သိမ်းဆည်းခဲ့ကြပြီး ယခုထက်ထိလည်း များစွာသော လော့ဂ်များကို ထိုနေရာ၌ပင် သိမ်းဆည်းထားဆဲ ဖြစ်သည်။ ပုံမှန်အားဖြင့် ပရိုဂရမ် တစ်ခုစီအတွက် ဖိုင် သို့မဟုတ် ဖိုဒါ တစ်ခုစီ ရှိတတ်သည်။ ၎င်းတို့ကို ရှာဖွေစစ်ဆေးရန် `grep` သို့မဟုတ် `less` ကို အသုံးပြုပါ။

`dmesg` command ကို အသုံးပြု၍ ကြည့်ရှုနိုင်သော ကာနယ် လော့ဂ် (kernel log) လည်း ရှိပါသည်။ ယခင်က ၎င်းကို plain-text ဖိုင်အဖြစ် ရယူနိုင်ခဲ့သော်လည်း ယခုအခါတွင်မူ ၎င်းကို ကြည့်ရှုနိုင်ရန် `dmesg` မှ တစ်ဆင့် သွားရောက်ရလေ့ ရှိသည်။

နောက်ဆုံးအနေဖြင့် သင်၏ log မက်ဆေ့ဂျ်များ အားလုံး အဓိက သွားရောက် စုစည်းသည့် "system log" ရှိပါသည်။ Linux စနစ် အားလုံးမဟုတ်သော်လည်း *အများစု* တွင် ထို log ကို နောက်ကွယ်၌ run နေသော ဝန်ဆောင်မှု (service) များအားလုံးကို ထိန်းချုပ်ပေးသည့် "system daemon" ဖြစ်သော `systemd` က စီမံခန့်ခွဲပေးပါသည် (ယခုအခါ နောက်ထပ် များစွာသော အရာများကိုပါ ထိန်းချုပ်ပေးထားပါသည်)။ သင်သည် root ဖြစ်ပါက သို့မဟုတ် `admin` သို့မဟုတ် `wheel` group များ၏ အဖွဲ့ဝင် ဖြစ်ပါက ထို log ကို အနည်းငယ် သုံးရခက်သော `journalctl` tool မှတစ်ဆင့် ရယူကြည့်ရှုနိုင်ပါသည်။

`journalctl` အတွက် အထူးသဖြင့် အောက်ပါ flag များကို သိရှိထားသင့်ပါသည်-

- `-u UNIT` : ပေးထားသော systemd service နှင့် သက်ဆိုင်သည့် မက်ဆေ့ဂျ်များကိုပဲ ဖော်ပြမည်
- `--full` : စာကြောင်းရှည်များကို ဖြတ်မပစ်ဘဲ (don't truncate) အပြည့်အစုံ ဖော်ပြမည်
- `-b` : နောက်ဆုံး စက်ပွင့်ခဲ့သည့်အချိန် (latest boot) မှ မက်ဆေ့ဂျ်များကိုသာ ဖော်ပြမည် (`-b -2` ကို လည်း ကြည့်ပါ)
- `-n100` : နောက်ဆုံး ပါဝင်သော entry ၁၀၀ ကိုသာ ဖော်ပြမည်

လက်ရှိ ဘာတွေ ဖြစ်ပျက်နေသလဲ။

တစ်ခုခု အမှားဖြစ်နေခဲ့လျှင် သို့မဟုတ် သင့်စနစ်တွင် လက်ရှိ ဘာတွေဖြစ်ပျက်နေသည်ကို သိရှိလိုရုံမျှဆိုလျှင် လက်ရှိ run နေသော စနစ်ကို စစ်ဆေးနိုင်သည့် တူးလ်အမြောက်အမြား ရှိပါသည်။

ပထမဦးစွာ သင့်စနစ်တွင် လက်ရှိ run နေသော ပရိုဆက်စ် (process) များ၏ စာရင်းဇယား အချက်အလက် (statistics) များစုံကို ပြသပေးသည့် `top` နှင့် ပိုမိုကောင်းမွန်အောင် ပြုလုပ်ထားသော `htop` တူးလ်များ ရှိပါသည်။ CPU အသုံးပြုမှု၊ memory အသုံးပြုမှု၊ process tree စသည်တို့ကို ပြသပေးသည်။ လမ်းဖြတ် ခလုတ် (shortcut) များစွာ ရှိသည့်အနက် `t` ခလုတ်သည် tree view ဖြင့် ကြည့်ရှုနိုင်ရန် အထူးပင် အသုံးဝင်ပါသည်။ Process tree ကို `pstree` (`+ -p` ထည့်သွင်းပါက PID များကိုပါ ပြသမည်) ဖြင့်လည်း ကြည့်ရှုနိုင်ပါသည်။ ထိုပရိုဂရမ်များ ဘာလုပ်နေသည်ကို သိရှိလိုပါက ၎င်းတို့၏ log ဖိုင်များကို တိုက်ရိုက် ကြည့်ရှု (tail) လေ့ရှိကြသည်။ ၎င်းအတွက် `journalctl -f` ၊ `dmesg -w` နှင့် `tail -f` တို့သည် သင့်အတွက် အလွန် အသုံးဝင်ပါလိမ့်မည်။

တစ်ခါတစ်ရံတွင် သင့်စနစ်၌ စုစုပေါင်း အသုံးပြုနေသော ရင်းမြစ် (resource) များနှင့် ပတ်သက်၍ ပိုမို အသေးစိတ် သိရှိလိုပေမည်။ **`dool`** (https://github.com/scottchiefbaker/dool) သည် အထူးပင် ကောင်းမွန်ပါသည်။ ၎င်းသည် I/O၊ networking၊ CPU အသုံးပြုမှု၊ context switch စသည့် သီးခြား စနစ်ခွဲ (subsystem) များစုံ၏ တိုက်ရိုက် ရင်းမြစ် တိုင်းတာချက်များ (real-time resource metrics) ကို ပြသပေးပါသည်။ `man dool` ကို စတင် လေ့လာနိုင်ပါသည်။

ဒစ်ခ်နေရာလွတ် (disk space) ကုန်သွားပါက သိရှိထားသင့်သည့် အဓိက တူးလ် (utility) နှစ်ခု ရှိပါသည်- `df` နှင့် `du` တို့ ဖြစ်ကြသည်။ `df` သည် သင့်စနစ်ရှိ partition များအားလုံး၏ အခြေအနေကို ပြသပေးပြီး (`-h` ထည့်သွင်း၍ စမ်းသပ်ကြည့်ပါ)၊ `du` သည်မူ သင်ပေးလိုက်သော ဖိုဒါများ (၎င်းတို့၏ ပါဝင်သည့် အကြောင်းအရာများ အပါအဝင်) ၏ အရွယ်အစားကို တိုင်းတာပေးပါသည် (`-h` နှင့် `-s` တို့ကိုလည်း ကြည့်ပါ)။

မည်သည့် ကွန်ရက် ချိတ်ဆက်မှု (network connection) များ ပွင့်နေသည်ကို ရှာဖွေကြည့်ရှုရန် `ss` ကို အသုံးပြုနိုင်ပါသည်။ `ss -t` သည် ပွင့်နေသော TCP ချိတ်ဆက်မှု အားလုံးကို ပြသပေးမည် ဖြစ်ပြီး၊ `ss -tl` သည် သင့်စနစ်တွင် နားထောင်နေသော (ဆိုလိုသည်မှာ စာဗာအဖြစ် အလုပ်လုပ်နေသော) Port (port) အားလုံးကို ပြသပေးမည် ဖြစ်သည်။ `-p` ထည့်သွင်းပါက ထိုချိတ်ဆက်မှုကို မည်သည့် process က အသုံးပြုနေသည်ကိုပါ ပြသပေးမည်ဖြစ်ပြီး `-n` ကမူ မူရင်း port နံပါတ်များကို သီးသန့် ပြသပေးမည် ဖြစ်ပါသည်။

စနစ် စီမံပြင်ဆင်ခြင်း (System configuration)

သင့်စနစ်ကို စီမံပြင်ဆင်ရန် (configure) နည်းလမ်း *များစွာ* ရှိသော်လည်း ကျွန်ုပ်တို့သည် အလွန်အသုံးများသော နည်းလမ်း နှစ်ခုဖြစ်သည့် ကွန်ရက် (networking) နှင့် ဝန်ဆောင်မှုများ (services) အကြောင်းကို လေ့လာသွားပါမည်။ သင့်စနစ်ရှိ application Manual page (man page) များတွင် ၎င်းတို့ကို မည်သို့ ပြင်ဆင်ရမည်ကို ဖော်ပြထားလေ့ရှိပြီး ပုံမှန်အားဖြင့် စနစ် စီမံပြင်ဆင်မှု ဖိုင်ဒါဖြစ်သော `/etc` ထဲရှိ ဖိုင်များကို ပြင်ဆင်ခြင်း ပါဝင်ပါသည်။

သင်၏ ကွန်ရက်ကို ပြင်ဆင်လိုပါက `ip` command ဖြင့် ပြုလုပ်နိုင်ပါသည်။ ၎င်း၏ အာဂူးမန့် (argument) များသည် အနည်းငယ် ဆန်းကြယ်သော ပုံစံရှိသော်လည်း `ip help command` သည် သင့်အား များစွာ ကူညီပေးနိုင်ပါသည်။ `ip addr` သည် သင်၏ network interface များနှင့် ၎င်းတို့ကို မည်သို့ ပြင်ဆင်ထားကြောင်း (IP address များ စသဖြင့်) သတင်းအချက်အလက်ကို ပြသပေးပြီး `ip route` ကမူ ကွန်ရက် အချက်အလက်များ (network traffic) ကို မတူညီသော ကွန်ရက် ဟို့စ် (network host) များထံ မည်သို့ လမ်းကြောင်းလွှဲထားကြောင်း ပြသပေးသည်။ ကွန်ရက် ပြဿနာများကို `ip` တူးလ် သီးသန့်ဖြင့်ပင် ဖြေရှင်းနိုင်လေ့ ရှိသည်။ ကြိုးမဲ့ ကွန်ရက် (wireless network) စီမံခန့်ခွဲရန်အတွက် `iw` လည်း ရှိပါသည်။ `ping` သည် မည်မျှအထိ အမှားအယွင်း ဖြစ်နေကြောင်း စစ်ဆေးရန် အလွန် အသုံးဝင်သော တူးလ်တစ်ခု ဖြစ်သည်။ Hostname တစ်ခု (google.com)၊ ပြင်ပ IP address တစ်ခု (1.1.1.1) နှင့် အတွင်းပိုင်း IP address တစ်ခု (192.168.1.1 သို့မဟုတ် default gateway) တို့ကို ping လုပ်ကြည့်ပါ။ သင့် DNS ဆက်တင်များ (hostname များကို IP address အဖြစ် ပြောင်းလဲခြင်း) ကို စစ်ဆေးရန် `/etc/resolv.conf` ကိုလည်း ဝင်ရောက် စစ်ဆေး ပြင်ဆင်နိုင်ပါသည်။

ဝန်ဆောင်မှုများကို ပြင်ဆင်ရန်အတွက် ယခုအခါ ကောင်းသည်ဖြစ်စေ၊ ဆိုးသည်ဖြစ်စေ `systemd` နှင့် ထိတွေ့ ဆက်ဆံရမည် ဖြစ်သည်။ သင့်စနစ်ရှိ ဝန်ဆောင်မှု အများစုတွင် systemd *unit* တစ်ခုကို သတ်မှတ်ပေးသည့် systemd service ဖိုင်တစ်ဖိုင် ရှိတတ်ကြသည်။ ထိုဖိုင်များသည် ဝန်ဆောင်မှုကို စတင်ချိန်တွင် မည်သည့် command ကို runရမည်၊ မည်သို့ ရပ်တန့်ရမည်၊ လော့ဂ်များကို မည်သည့်နေရာတွင် သိမ်းဆည်းရမည် စသည်တို့ကို သတ်မှတ်ပေးသည်။ ၎င်းတို့ကို ဖတ်ရှုရသည်မှာ ခက်ခဲလေ့မရှိဘဲ အများစုကို `/usr/lib/systemd/system/` တွင် တွေ့ရှိနိုင်ပါသည်။ မိမိပိုင် ဖိုင်များကိုမူ `/etc/systemd/system` တွင် သတ်မှတ်နိုင်ပါသည်။

သင် စီမံလိုသည့် systemd service ကို သတ်မှတ်ပြီးပါက ၎င်းကို ထိန်းချုပ်ရန် `systemctl` command ကို အသုံးပြုနိုင်ပါသည်။ `systemctl enable UNIT` သည် ထို service ကို စက်စတင်ပွင့်ချိန် (boot) တွင် runရန် ပြင်ဆင်ပေးမည် ဖြစ်ပြီး (`disable` က ပြန်လည် ဖယ်ရှားပေးပါမည်)၊ `start` ၊ `stop` နှင့် `restart` တို့သည်လည်း မျှော်လင့်ထားသည့်အတိုင်း အလုပ်လုပ်မည် ဖြစ်သည်။ တစ်ခုခု အမှားအယွင်း ဖြစ်ပါက systemd က အသိပေးမည်ဖြစ်ပြီး ၎င်း application ၏ log ကို ကြည့်ရန် `journalctl -u UNIT` ကို အသုံးပြုနိုင်ပါသည်။ သင့်စနစ်၏ ဝန်ဆောင်မှုများ မည်သို့ အလုပ်လုပ်နေသည်ကို ကြည့်ရန် `systemctl`

`status` ကိုလည်း အသုံးပြုနိုင်ပါသည်။ စက်စတင်ပွင့်ချိန် (boot) အချိန်ကြာမြင့်နေပါက ၎င်းသည် နှေးကွေးနေသော ဝန်ဆောင်မှု အချို့ကြောင့် ဖြစ်နိုင်ပြီး မည်သည့် ဝန်ဆောင်မှုများဖြစ်သည်ကို ရှာဖွေရန် `systemd-analyze` (`blame` ဖြင့် စမ်းသပ်ကြည့်ပါ) ကို အသုံးပြုနိုင်ပါသည်။

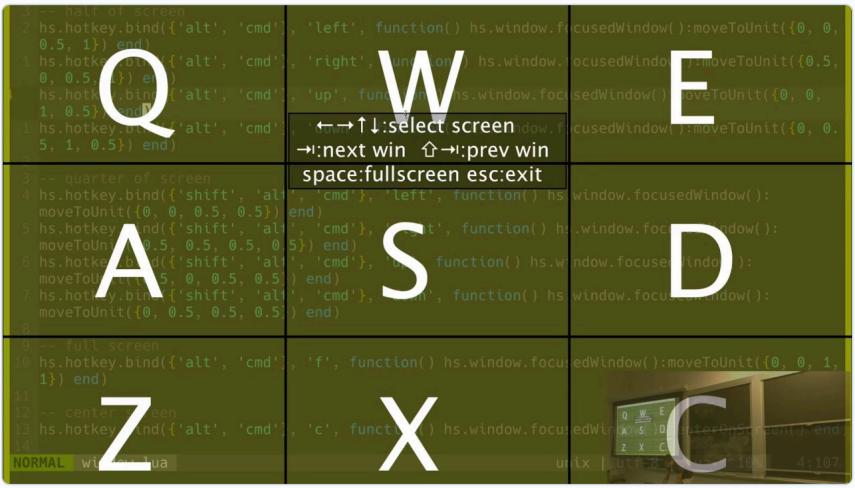
🔗 External Links & References (ပြင်ပ လင့်ခ်များနှင့် ကိုးကားချက်များ)

Resource / Description	URL	Micro QR
1. `dool`	https://github.com/scottchiefbaker/dool	

OS Customization

▶ LECTURE VIDEO REFERENCE

OS Customization



Scan the QR code below using your mobile camera or click the link to watch this full lecture online:

Direct Online Link:

https://www.youtube.com/watch?v=epSRVqQzeDo

သင့် operating system ၏ settings မီနူးများတွင် ပါရှိသည်များအပြင် မိမိ၏ OS ကို ပိုမိုစိတ်ကြိုက်ပြင်ဆင်နိုင်သော အရာများစွာ ရှိပါသည်။

ကီးဘုတ် ခလုတ်များ ပြန်လည်သတ်မှတ်ခြင်း (Keyboard remapping)

သင့်ကီးဘုတ်တွင် သင်အသုံးသိပ်မပြုသည့် ခလုတ်များ ရှိနေနိုင်ပါသည်။ ထိုသို့ အသုံးမဝင်သော ခလုတ်များ ရှိနေမည့်အစား ၎င်းတို့ကို အသုံးဝင်သော လုပ်ဆောင်ချက်များ ပြုလုပ်နိုင်ရန် ပြန်လည်သတ်မှတ်ပေးနိုင်ပါသည် (remap ပြုလုပ်နိုင်ပါသည်။)

အခြားခလုတ်များဆီသို့ ပြန်လည်သတ်မှတ်ခြင်း (Remapping to other keys)

အရိုးရှင်းဆုံး အရာမှာ ခလုတ်တစ်ခုကို အခြားခလုတ်တစ်ခုအဖြစ် ပြန်လည်သတ်မှတ်ပေးခြင်း ဖြစ်ပါသည်။ ဥပမာအားဖြင့် သင်သည် Caps Lock ခလုတ်ကို သိပ်မသုံးလျှင် ၎င်းကို ပိုမိုအသုံးဝင်သော ခလုတ်တစ်ခုခု အဖြစ် ပြောင်းလဲသတ်မှတ်နိုင်ပါသည်။ ဥပမာ သင်သည် Vim အသုံးပြုသူတစ်ဦးဖြစ်ပါက Caps Lock ခလုတ်ကို Escape ခလုတ်အဖြစ် ပြန်လည်သတ်မှတ်လိုပေမည်။

macOS တွင် System Preferences ရှိ Keyboard settings ထဲမှတစ်ဆင့် ခလုတ်အချို့ကို ပြန်လည်သတ်မှတ်နိုင်ပြီး ပိုမိုရှုပ်ထွေးသော သတ်မှတ်မှုများအတွက် သီးသန့်ဆော့ဖ်ဝဲလ် လိုအပ်ပါသည်။

စိတ်ကြိုက် command များဆီသို့ ပြန်လည်သတ်မှတ်ခြင်း (Remapping to arbitrary commands)

ခလုတ်တစ်ခုကို အခြားခလုတ်တစ်ခုအဖြစ်သာ မကဘဲ ခလုတ်များ (သို့မဟုတ် ခလုတ်တွဲများ) ကို မိမိစိတ်ကြိုက် command များနှင့် တွဲဖက်သတ်မှတ်ပေးနိုင်သော tool များလည်း ရှိပါသည်။ ဥပမာအားဖြင့် `command-shift-t` နှိပ်လိုက်ပါက terminal window အသစ်တစ်ခု ဖွင့်လှစ်ပေးနိုင်ရန် သတ်မှတ်နိုင်ပါသည်။

ဝှက်ထားသော OS Settings များကို စိတ်ကြိုက်ပြင်ဆင်ခြင်း (Customizing hidden OS settings)

macOS

macOS တွင် `defaults` command မှတစ်ဆင့် အသုံးဝင်သော setting အများအပြားကို ပြင်ဆင်နိုင်ရန် ပံ့ပိုးပေးထားပါသည်။ ဥပမာအားဖြင့် ဝှက်ထားသော application များ၏ Dock icon များကို အလင်းစိမ့်ဝင်မြင်နိုင်သော (translucent) ပုံစံ ဖြစ်စေရန် ပြုလုပ်နိုင်ပါသည်-

```
defaults write com.apple.dock showhidden -bool true
```

ဖြစ်နိုင်ခြေရှိသော setting များ အားလုံးပါဝင်သည့် စာရင်းတစ်ခုတည်း ရနိုင်မည်မဟုတ်သော်လည်း သီးခြားစိတ်ကြိုက် ပြင်ဆင်မှုများ၏ စာရင်းများကို အွန်လိုင်းတွင် ရှာဖွေနိုင်ပါသည်၊ ဥပမာအားဖြင့် Mathias Bynens ၏ **.macos** (https://github.com/mathiasbynens/dotfiles/blob/master/.macos) ကဲ့သို့ ဖြစ်ပါသည်။

Window စီမံခန့်ခွဲခြင်း (Window management)

Tiling window management (ဝင်းဒိုးများကို မျက်နှာပြင်ပေါ်တွင် နေရာယူစီစဉ်ခြင်း)

Tiling window management (https://en.wikipedia.org/wiki/Tiling_window_manager) ဆိုသည်မှာ window များကို ထပ်မနေသော frame များအဖြစ် စီစဉ်ပေးသည့် window စီမံခန့်ခွဲမှု နည်းလမ်းတစ်ခု ဖြစ်ပါသည်။ သင်သည် Linux အခြေခံ operating system ကို အသုံးပြုနေပါက tiling window manager တစ်ခုကို ထည့်သွင်းအသုံးပြုနိုင်ပြီး Windows သို့မဟုတ် macOS ကဲ့သို့သော OS များကို အသုံးပြုနေပါက ထိုသို့လုပ်ဆောင်ချက်မျိုး ရရှိစေမည့် application များကို ထည့်သွင်းနိုင်ပါသည်။

Screen စီမံခန့်ခွဲခြင်း (Screen management)

မျက်နှာပြင်များအကြား window များကို ရွှေ့ပြောင်း ထိန်းချုပ်နိုင်ရန် ကီးဘုတ် shortcut များကို သတ်မှတ်ထားနိုင်ပါသည်။

Layout များ (Layouts)

မျက်နှာပြင်တစ်ခုပေါ်တွင် window များကို သီးခြားနေရာချထားလေ့ရှိပါက ထို layout ကို ကိုယ်တိုင် ကိုယ်ကျလိုက်လံနေရာချနေမည့်အစား script ရေးသားထားနိုင်ပြီး layout အသစ်တစ်ခု ဖန်တီးခြင်းကို လွယ်ကူလျင်မြန်စွာ ပြုလုပ်နိုင်မည် ဖြစ်ပါသည်။

လေ့လာရန် သယံဇာတများနှင့် Tool များ (Resources)

- **Hammerspoon** (https://www.hammerspoon.org/) - macOS desktop များကို အလိုအလျောက် လုပ်ဆောင်ပေးသည့် tool
- **Rectangle** (https://rectangleapp.com/) - macOS window manager
- **Karabiner** (https://karabiner-elements.pqrs.org/) - ဆန်းသစ်သော macOS keyboard remapping tool
- **r/unixporn** (https://www.reddit.com/r/unixporn/) - လူအများ၏ ဆန်းသစ်လှပသော configuration များ၏ screenshot များနှင့် မှတ်တမ်းများ

လေ့ကျင့်ခန်းများ (Exercises)

1. သင့် Caps Lock ခလုတ်ကို သင် ပိုမိုအသုံးပြုသည့် ခလုတ်တစ်ခုခုဆီသို့ (ဥပမာ Escape၊ Ctrl သို့မဟုတ် Backspace ကဲ့သို့သော) မည်သို့ ပြန်လည်သတ်မှတ်ရမည်ကို လေ့လာရှာဖွေပါ။
2. Terminal window အသစ် သို့မဟုတ် browser window အသစ် ဖွင့်လှစ်ရန် စိတ်ကြိုက် global keyboard shortcut တစ်ခု ပြုလုပ်ပါ။

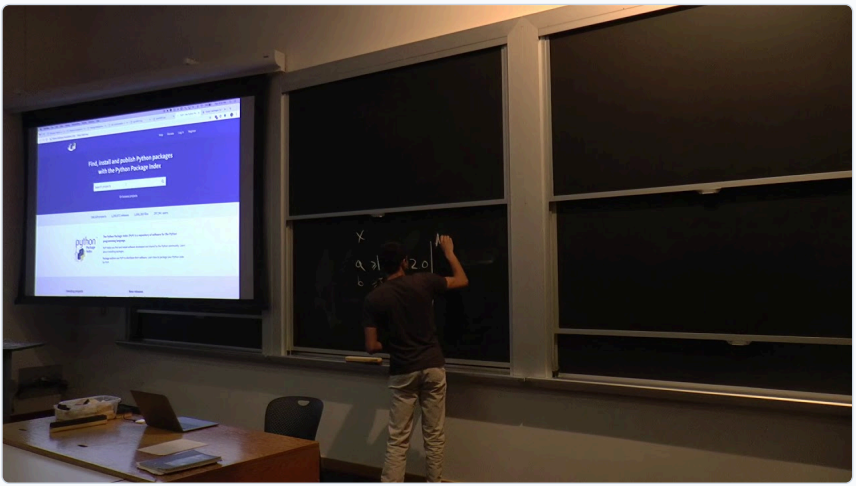
🔗 External Links & References (ပြင်ပ လင့်ခ်များနှင့် ကိုးကားချက်များ)

Resource / Description	URL	Micro QR
1. .macos	https://github.com/mathiasbynens/dotfiles/blob/master/.macos	
2. Tiling window management	https://en.wikipedia.org/wiki/Tiling_window_manager	
3. Hammerspoon	https://www.hammerspoon.org/	
4. Rectangle	https://rectangleapp.com/	
5. Karabiner	https://karabiner-elements.pqrs.org/	
6. r/unixporn	https://www.reddit.com/r/unixporn/	

Package Management နှင့် Dependency Management

▶ LECTURE VIDEO REFERENCE

Package Management နှင့် Dependency Management



Scan the QR code below using your mobile camera or click the link to watch this full lecture online:

Direct Online Link:

https://www.youtube.com/watch?v=tgvt473T8xA

ဆော့ဖ်ဝဲလ်များသည် အခြားသော ဆော့ဖ်ဝဲလ် (သို့မဟုတ် ဆော့ဖ်ဝဲလ်အစုအဝေး) များအပေါ် အခြေခံ၍ တည်ဆောက်လေ့ရှိသဖြင့် dependency စီမံခန့်ခွဲမှု (dependency management) ကို မဖြစ်မနေ လိုအပ်လာပါသည်။

Package/dependency စီမံခန့်ခွဲသည့် ပရိုဂရမ်များသည် ပရိုဂရမ်းမင်းဘာသာစကားအလိုက် သီးခြားစီ ကွဲပြားကြသော်လည်း အများစုမှာ ဘုံသဘောတရားအယူအဆများကို မျှဝေအသုံးပြုကြပါသည်။

Package repository များ

Package များကို *package repository များ* တွင် သိမ်းဆည်းထားရှိကြပါသည်။ ပရိုဂရမ်းမင်းဘာသာစကား အမျိုးအစားအလိုက် မတူညီသော repository များရှိကြပြီး (အချို့သော ဘာသာစကားတစ်ခုတည်းအတွက်ပင် repository အများအပြား ရှိနိုင်သည်)၊ ဥပမာအားဖြင့် Python အတွက် **PyPI** (https://pypi.org/)၊ Ruby အတွက် **RubyGems** (https://rubygems.org/) နှင့် Rust အတွက် **crates.io** (https://crates.io/) တို့ဖြစ်ကြပါသည်။ ယင်းတို့သည် အများအားဖြင့် package တစ်ခု၏ မူကွဲ (version) အားလုံးအတွက် ဆော့ဖ်ဝဲလ်များ (source code များ သို့မဟုတ် အချို့သော သီးခြား platform များအတွက် ကြိုတင် compile လုပ်ထားသည့် binary များ) ကို သိမ်းဆည်းပေးထားကြပါသည်။

Semantic versioning

ဆော့ဖ်ဝဲလ်များသည် အချိန်နှင့်အမျှ ပြောင်းလဲတိုးတက်လာသောကြောင့် ဆော့ဖ်ဝဲလ် version များကို ရည်ညွှန်းညွှန်းဆိုနိုင်မည့် နည်းလမ်းတစ်ခု လိုအပ်ပါသည်။ အစဉ်လိုက်နံပါတ် သို့မဟုတ် commit hash ဖြင့် ရည်ညွှန်းခြင်းသည် ရိုးရှင်းသော နည်းလမ်းအချို့ ဖြစ်နိုင်သော်လည်း၊ version နံပါတ်များကို အသုံးပြုခြင်းဖြင့် ပိုမိုပြည့်စုံသော အချက်အလက်များကို ပေးပို့နိုင်သည့် ပိုမိုကောင်းမွန်သော နည်းလမ်းကို အသုံးပြုနိုင်ပါသည်။

နည်းလမ်းမျိုးစုံ ရှိသည့်အနက် ရေပန်းစားသော နည်းလမ်းတစ်ခုမှာ **Semantic Versioning** (https://semver.org/) ဖြစ်ပါသည်။

```
x.y.z
^ ^ ^
| | +- patch
| +--- minor
+----- major
```

ရှေ့နောက် ကိုက်ညီမှုမရှိသော (incompatible) API ပြောင်းလဲမှုများကို ပြုလုပ်သည့်အခါ **major** version နံပါတ်ကို တိုးပေးပါ။

နောက်ပြန် ကိုက်ညီမှုရှိသော (backward-compatible) နည်းလမ်းဖြင့် လုပ်ဆောင်ချက်အသစ်များကို ထည့်သွင်းသည့်အခါ **minor** version နံပါတ်ကို တိုးပေးပါ။

နောက်ပြန် ကိုက်ညီမှုရှိသော bug ပြင်ဆင်မှုများကို ပြုလုပ်သည့်အခါ **patch** version နံပါတ်ကို တိုးပေးပါ။

ဥပမာအားဖြင့်၊ သင်သည် အချို့သော ဆော့ဖ်ဝဲလ်၏ `v1.2.0` တွင် စတင်မိတ်ဆက်ခဲ့သည့် အင်္ဂါရုပ်တစ်ခုအပေါ် မှီခိုနေရပါက၊ minor version `x >= 2` နှင့် မည်သည့် patch version `y` အတွက်မဆို `v1.x.y` ကို တပ်ဆင်နိုင်ပါသည်။ major version `1` ကို တပ်ဆင်ရန် လိုအပ်ပြီး (အကြောင်းမှာ `2` သည် နောက်ပြန်ကိုက်ညီမှုမရှိသော ပြောင်းလဲမှုများကို သယ်ဆောင်လာနိုင်သောကြောင့်ဖြစ်သည်)၊ minor version `>= 2` ကို တပ်ဆင်ရန် လိုအပ်ပါသည် (အကြောင်းမှာ ထို minor version တွင် စတင်မိတ်ဆက်ခဲ့သော အင်္ဂါရုပ်အပေါ် မှီခိုနေသောကြောင့်ဖြစ်သည်)။ နောက်ပိုင်းထွက် ပိုမိုသစ်လွင်သော minor version သို့မဟုတ် patch version မည်သည့်အရာကိုမဆို အသုံးပြုနိုင်သည်၊ အကြောင်းမှာ ယင်းတို့သည် နောက်ပြန် ကိုက်ညီမှုမရှိသော ပြောင်းလဲမှုများကို သယ်ဆောင်လာမည် မဟုတ်သောကြောင့်ဖြစ်ပါသည်။

Lock file များ

Version များကို သတ်မှတ်ပေးသည့်အပြင်၊ မသမာသော မသမာပြုပြင်မှုများကို ကာကွယ်ရန် dependency ၏ *အကြောင်းအရာများ (contents)* ပြောင်းလဲခြင်း မရှိကြောင်း အတင်းအကျပ် သတ်မှတ်ပေးနိုင်ပါက ပိုမို ကောင်းမွန်ပါသည်။ အချို့သော tool များသည် package တပ်ဆင်ချိန်တွင် စစ်ဆေးမည့် dependency ၏ cryptographic hash များကို (version များနှင့်အတူ) သတ်မှတ်ရန် *lock file များ* ကို အသုံးပြုကြပါသည်။

Version များကို သတ်မှတ်ခြင်း

Tool များသည် Version များကို အောက်ပါအတိုင်း နည်းလမ်းမျိုးစုံဖြင့် သတ်မှတ်နိုင်ရန် ဖန်တီးပေးထားလေ့ရှိကြပါသည် -

- တိကျသော version၊ ဥပမာ `2.3.12`
- အနည်းဆုံး major version၊ ဥပမာ `>= 2`
- သီးခြား major version နှင့် အနည်းဆုံး patch version၊ ဥပမာ `>= 2.3, <3.0`

တိကျသော version တစ်ခုကို သတ်မှတ်ခြင်းသည် တပ်ဆင်ထားသော dependency များပေါ်မူတည်၍ ကွဲပြားသော မူမမှန်မှုများကို ရှောင်ရှားနိုင်သည့် အကျိုးကျေးဇူး ရှိပါသည် (dependency အားလုံးသည် semver စံနှုန်းကို တိကျစွာ လိုက်နာပါက ဤသို့မဖြစ်သင့်သော်လည်း တစ်ခါတစ်ရံ လူတို့ အမှားပြုလုပ်တတ်ကြပါသည်)။ အနည်းဆုံး လိုအပ်ချက်ကို သတ်မှတ်ခြင်းသည် bug ပြင်ဆင်မှုများကို တပ်ဆင်ခွင့်ပြုသည့် အကျိုးကျေးဇူး ရှိပါသည် (ဥပမာ- patch မူကွဲမြင့်တင်မှုများ)။

Dependency resolution

Package manager များသည် dependency လိုအပ်ချက်များကို ပြည့်မီစေရန် အမျိုးအစားစုံလင်သော dependency resolution algorithm များကို အသုံးပြုကြပါသည်။ ရှုပ်ထွေးသော dependency များရှိသည့် အခါ (ဥပမာ- package တစ်ခုကို ထိပ်တန်း dependency အများအပြားက သွယ်ဝိုက်၍ မှီခိုနေနိုင်ပြီး မတူညီသော version များကို လိုအပ်နေနိုင်ပါသည်) ဤသည်မှာ စိန်ခေါ်မှုများပြားလာလေ့ ရှိပါသည်။ မတူညီသော package manager များသည် ၎င်းတို့၏ dependency resolution တွင် ဆန်းသစ်ရှုပ်ထွေးမှု အဆင့်အတန်းကွဲပြားကြသော်လည်း၊ ၎င်းသည် သတိပြုရမည့် အရာတစ်ခုဖြစ်ပါသည်- သင်သည် dependency များကို debug လုပ်နေပါက ဤအကြောင်းကို နားလည်ထားရန် လိုအပ်နိုင်ပါသည်။

Virtual environment များ

သင်သည် ဆော့ဖ်ဝဲလ် ပရောဂျက် အများအပြားကို ရေးသားထုတ်လုပ်နေပါက၊ ၎င်းတို့သည် သီးခြားဆော့ဖ်ဝဲလ်တစ်ခု၏ မတူညီသော version များကို မှီခိုနေနိုင်ပါသည်။ အချို့သော အခြေအနေများတွင် သင်၏ build tool က ဤပြဿနာကို အလိုအလျောက် ဖြေရှင်းပေးပါလိမ့်မည် (ဥပမာ- static binary တစ်ခုအဖြစ် တည်ဆောက်ခြင်းဖြင့်)။

အခြားသော build tool များနှင့် ပရိုဂရမ်းမင်းဘာသာစကားများအတွက် နည်းလမ်းတစ်ခုမှာ virtual environment များကို အသုံးပြု၍ စီမံခန့်ခွဲခြင်းဖြစ်ပါသည် (ဥပမာ- Python အတွက် **virtualenv** (https://docs.python-guide.org/dev/virtualenvs/) tool ဖြင့်)။ Dependency များကို စနစ်တစ်ခုလုံး (system-wide) တွင် တပ်ဆင်မည့်အစား၊ virtual environment တစ်ခုအတွင်း ပရောဂျက်တစ်ခုချင်းစီအလိုက် တပ်ဆင်နိုင်ပြီး၊ သီးခြားပရောဂျက်တစ်ခုတွင် အလုပ်လုပ်နေချိန်၌ သင်အသုံးပြုလိုသော virtual environment ကို *activate* ပြုလုပ် အသုံးပြုနိုင်ပါသည်။

Vendoring

Dependency စီမံခန့်ခွဲမှုအတွက် သီးခြားကွဲပြားသော အခြားနည်းလမ်းတစ်ခုမှာ *vendoring* ဖြစ်ပါသည်။ ဆော့ဖ်ဝဲလ်ကို ရယူရန် dependency manager သို့မဟုတ် build tool ကို အသုံးပြုမည့်အစား၊ dependency ၏ source code တစ်ခုလုံးကို သင်၏ ဆော့ဖ်ဝဲလ် repository ထဲသို့ ကူးယူထည့်သွင်းလိုက်ခြင်း ဖြစ်ပါသည်။ ဤသည်မှာ သင်သည် dependency ၏ တူညီသော version ပေါ်တွင် အမြဲတမ်း တည်ဆောက်နေနိုင်ပြီး package repository ပေါ်တွင် မှီခိုနေရန် မလိုတော့သည့် အကျိုးကျေးဇူး ရှိသော်လည်း၊ dependency များကို အဆင့်မြှင့်တင်ရန်အတွက် ပိုမိုအားထုတ် ကြိုးပမ်းရပါသည်။

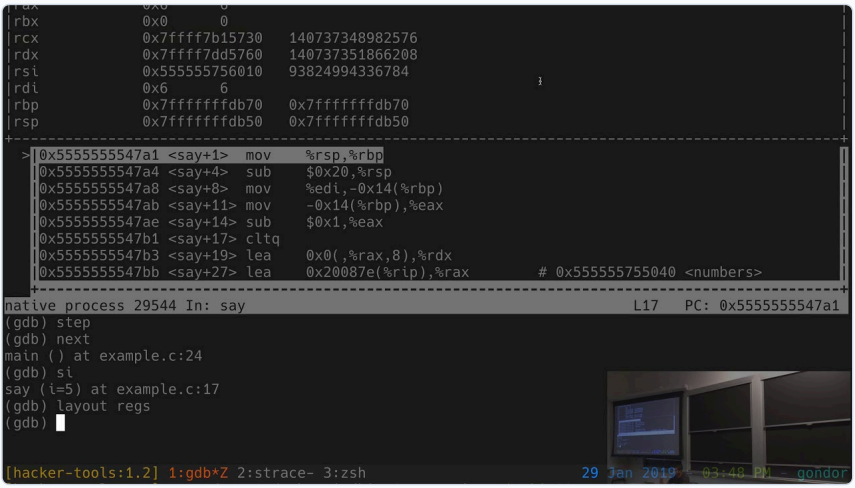
🔗 External Links & References (ပြင်ပ လင့်ခ်များနှင့် ကိုးကားချက်များ)

Resource / Description	URL	Micro QR
1. PyPI	https://pypi.org/	
2. RubyGems	https://rubygems.org/	
3. crates.io	https://crates.io/	
4. Semantic Versioning	https://semver.org/	
5. virtualenv	https://docs.python-guide.org/dev/virtualenvs/	

Program Introspection

▶ LECTURE VIDEO REFERENCE

Program Introspection



Scan the QR code below using your mobile camera or click the link to watch this full lecture online:

Direct Online Link:

https://www.youtube.com/watch?v=74MhV-7hYzg

Debugging

printf-debugging ပြုလုပ်ခြင်းက လုံလောက်မှု မရှိတော့သည့်အခါ debugger ကို အသုံးပြုပါ။

Debugger များသည် ပရိုဂရမ်၏ အလုပ်လုပ်ဆောင်မှု (execution) နှင့် တိုက်ရိုက် ထိတွေ့ဆက်ဆံ လုပ်ဆောင်နိုင်စေပြီး အောက်ပါ အချက်များကို ပြုလုပ်နိုင်စေပါသည်။

- သတ်မှတ်ထားသော စာကြောင်းတစ်ကြောင်းသို့ ရောက်ရှိသည့်အခါ ပရိုဂရမ် အလုပ်လုပ်နေမှုကို ရပ်တန့်ခြင်း
- ပရိုဂရမ်ကို တစ်ကြောင်းချင်းစီ (single-step) အဆင့်လိုက် ပတ်လည်စစ်ဆေး မောင်းနှင်ခြင်း
- Variable များ၏ တန်ဖိုးများကို စစ်ဆေးခြင်း
- အခြားသော အဆင့်မြင့် လုပ်ဆောင်ချက်များစွာ ပြုလုပ်နိုင်ခြင်း

GDB/LLDB

GDB (https://www.gnu.org/software/gdb/) နှင့် **LLDB** (https://lldb.llvm.org/)။ C နှင့် ဆင်တူသော ဘာသာစကား အမြောက်အများကို ထောက်ပံ့ပေးထားပါသည်။

example.c ကို ကြည့်ကြပါစို့။ Debug flags ဖြင့် compile လုပ်ပါ: `gcc -g -o example example.c` ။

GDB ကို ဖွင့်ပါ:

```
gdb example
```

အချို့သော command များ:

- `run`
- `b {name of function}` - breakpoint တစ်ခု သတ်မှတ်ရန်
- `b {file}:{line}` - breakpoint တစ်ခု သတ်မှတ်ရန်
- `c` - ဆက်လက် အလုပ်လုပ်ရန် (continue)
- `step` / `next` / `finish` - step in / step over / step out ပြုလုပ်ရန်
- `p {variable}` - variable ၏ တန်ဖိုးကို ထုတ်ပြရန်

- `watch {expression}` - expression ၏ တန်ဖိုး ပြောင်းလဲသွားသည့်အခါ အလိုအလျောက် သတိပေးရပ်တန့်မည့် watchpoint တစ်ခု သတ်မှတ်ရန်
- `rwatch {expression}` - တန်ဖိုးကို ဖတ်ရှုသည့်အခါ အလိုအလျောက် သတိပေးရပ်တန့်မည့် watchpoint တစ်ခု သတ်မှတ်ရန်
- `layout`

PDB

PDB (https://docs.python.org/3/library/pdb.html) သည် Python debugger ဖြစ်ပါသည်။

PDB ထဲသို့ ရောက်ရှိလိုသည့် နေရာတွင် `import pdb; pdb.set_trace()` ကို ထည့်သွင်းပါ။ ၎င်းသည် အခြေခံအားဖြင့် debugger (GDB ကဲ့သို့) နှင့် Python shell တို့ကို ပေါင်းစပ်ထားသော hybrid စနစ်တစ်ခု ဖြစ်ပါသည်။

Web browser Developer Tools

နောက်ထပ် debugger အမျိုးအစား တစ်ခု ဖြစ်ပြီး ယခုတစ်ကြိမ်တွင်မူ graphical interface ပါဝင်ပါသည်။

strace

ပရိုဂရမ်တစ်ခုမှ ပြုလုပ်သော system call များကို လေ့လာစောင့်ကြည့်ခြင်း: `strace {program}` ။

Profiling

Profiling အမျိုးအစားများ: CPU, memory အစရှိသည်တို့ ဖြစ်ကြသည်။

အရိုးရှင်းဆုံး profiler: `time` ။

Go

Test code ကို CPU profiler ဖြင့် မောင်းနှင်ပါ: `go test -cpuprofile=cpu.out`

Profile ကို ဆန်းစစ်သုံးသပ်ပါ: `go tool pprof -web cpu.out`

Test code ကို Memory profiler ဖြင့် မောင်းနှင်ပါ: `go test -memprofile=mem.out`

Profile ကို ဆန်းစစ်သုံးသပ်ပါ: `go tool pprof -web mem.out`

Perf

အခြေခံ စွမ်းဆောင်ရည် စာရင်းအင်းများ: `perf stat {command}`

ပရိုဂရမ်ကို profiler ဖြင့် မောင်းနှင်ပါ: `perf record {command}`

Profile ကို ဆန်းစစ်သုံးသပ်ပါ: `perf report`

🔗 External Links & References (ပြင်ပ လင့်ခ်များနှင့် ကိုးကားချက်များ)

Resource / Description	URL	Micro QR
1. GDB	https://www.gnu.org/software/gdb/	
2. LLDB	https://lldb.llvm.org/	
3. PDB	https://docs.python.org/3/library/pdb.html	

Remote Machines

▶ LECTURE VIDEO REFERENCE

Remote Machines

Scan the QR code below using your mobile camera or click the link to watch this full lecture online:

Direct Online Link:

https://www.youtube.com/watch?v=X5c2Y8BCowM

Programmer များအနေဖြင့် မိမိတို့၏ နေ့စဉ်လုပ်ငန်းခွင်တွင် remote server များကို အသုံးပြုလာကြခြင်းသည် ပိုမို ခေတ်စားလာခဲ့ပါသည်။ အကယ်၍ သင်သည် backend ဆော့ဖ်ဝဲလ်များကို ဖြန့်ကျက်ရန် (deploy ပြုလုပ်ရန်) သို့မဟုတ် ပိုမိုမြင့်မားသော တွက်ချက်မှုစွမ်းရည် (computational capabilities) ရှိသည့် server တစ်ခု လိုအပ်၍ remote server များကို အသုံးပြုရန် လိုအပ်ပါက၊ နောက်ဆုံးတွင် သင်သည် Secure Shell (SSH) ကို အသုံးပြုရမည် ဖြစ်ပါသည်။ ဤသင်တန်းတွင် ဖော်ပြခဲ့သော tool အများစုကဲ့သို့ပင် SSH သည်

အလွန်အမင်း စိတ်ကြိုက် ပြင်ဆင်သတ်မှတ်နိုင်စွမ်း (highly configurable) ရှိသဖြင့် ယင်းအကြောင်းကို လေ့လာသင်ယူရကျိုး နပ်ပါသည်။

Command များကို မောင်းနှင်ခြင်း

`ssh` ၏ မကြာခဏ သတိမမူမိတတ်ကြသော feature တစ်ခုမှာ command များကို တိုက်ရိုက် မောင်းနှင်နိုင်သည့် (run နိုင်သည့်) စွမ်းဆောင်ရည် ဖြစ်ပါသည်။

- `ssh foobar@server ls` သည် foobar ၏ home folder အတွင်း၌ ls ကို မောင်းနှင်မည် ဖြစ်သည်
- ယင်းသည် pipe များနှင့်လည်း အလုပ်လုပ်သောကြောင့် `ssh foobar@server ls | grep PATTERN` သည် remote မှထွက်လာသော `ls` output ကို local တွင် grep လုပ်မည်ဖြစ်ပြီး၊ `ls | ssh foobar@server grep PATTERN` သည် local မှထွက်လာသော `ls` output ကို remote တွင် grep လုပ်မည် ဖြစ်သည်။

SSH Key များ

Key အခြေပြု အတည်ပြုစစ်ဆေးခြင်း (Key-based authentication) သည် လျှို့ဝှက် private key ကို ထုတ်ဖော်ပြသခြင်း မရှိဘဲ client တွင် ထို private key ရှိကြောင်း server ထံ သက်သေပြရန် public-key cryptography ကို အသုံးပြုထားခြင်း ဖြစ်ပါသည်။ ဤနည်းဖြင့် သင်သည် အကြိမ်တိုင်းတွင် သင်၏ password ကို ပြန်လည် ရိုက်ထည့်ရန် မလိုအပ်တော့ပါ။ သို့သော်လည်း private key (ဥပမာ `~/.ssh/id_rsa`) သည် အမှန်တကယ်အားဖြင့် သင်၏ password ပင် ဖြစ်သောကြောင့် ထိုအတိုင်း ဂရုစိုက် စီမံရမည် ဖြစ်သည်။

- **Key ထုတ်လုပ်ခြင်း (Key generation)**: Key အတွဲတစ်ခု ထုတ်လုပ်ရန် `ssh-keygen -t rsa -b 4096` ကို ရိုးရှင်းစွာ runနိုင်ပါသည်။ အကယ်၍ သင်သည် passphrase တစ်ခုကို မရွေးချယ်ပါက သင်၏ private key ကို ရရှိသွားသူ မည်သူမဆို ခွင့်ပြုချက်ရထားသော server များကို ဝင်ရောက် အသုံးပြုနိုင်မည် ဖြစ်သဖြင့် passphrase တစ်ခု ရွေးချယ်ပြီး shell session များကို စီမံခန့်ခွဲရန် `ssh-agent` ကို အသုံးပြုရန် အကြံပြုပါသည်။

အကယ်၍ သင်သည် SSH key များကို အသုံးပြု၍ GitHub သို့ push ပြုလုပ်ရန် ပြင်ဆင်ထားပြီးပါက [ဤ နေရာတွင်](https://help.github.com/articles/connecting-to-github-with-ssh/) ဖော်ပြထားသော အဆင့်များကို လုပ်ဆောင်ခဲ့ပြီးဖြစ်ကာ တရားဝင် key အတွဲတစ်ခု ရှိပြီးသား ဖြစ်နိုင်ပါသည်။ သင့်တွင် passphrase ရှိမရှိ စစ်ဆေးရန်နှင့် ယင်းကို အတည်ပြုရန်အတွက် `ssh-keygen -y -f /path/to/key` ကို runနိုင်ပါသည်။

- **Key အခြေပြု အတည်ပြုစစ်ဆေးခြင်း (Key based authentication)**: `ssh` သည် မည်သည့် client များကို ဝင်ရောက်ခွင့်ပြုရမည်ကို ဆုံးဖြတ်ရန် `.ssh/authorized_keys` ကို ကြည့်ရှုမည် ဖြစ်သည်။

Public key တစ်ခုကို အခြားဘက်သို့ ကူးယူရန်အတွက် အောက်ပါအတိုင်း အသုံးပြုနိုင်ပါသည် -

```
cat .ssh/id_dsa.pub | ssh foobar@remote 'cat >> ~/.ssh/authorized_keys'
```

ရရှိနိုင်သည့် နေရာများတွင် `ssh-copy-id` ကို အသုံးပြု၍ ပိုမို ရိုးရှင်းသော ဖြေရှင်းချက်ကို ရရှိနိုင်ပါသည်။

```
ssh-copy-id -i .ssh/id_dsa.pub foobar@remote
```

SSH မှတစ်ဆင့် ဖိုင်များကို ကူးယူခြင်း

SSH မှတစ်ဆင့် ဖိုင်များကို ကူးယူရန် နည်းလမ်းများစွာ ရှိပါသည် -

- **`ssh+tee`** : အရိုးရှင်းဆုံး နည်းလမ်းမှာ `cat localfile | ssh remote_server tee serverfile` ကို ပြုလုပ်ခြင်းဖြင့် `ssh` command execution နှင့် stdin input ကို အသုံးပြုခြင်း ဖြစ်ပါသည်။
- **`scp`** : ဖိုင်များ/directory ပမာဏ အများအပြားကို ကူးယူသည့်အခါ secure copy `scp` command သည် path များအပေါ် အဆင့်ဆင့် လွယ်ကူစွာ သွားရောက်နိုင်သောကြောင့် (recurse ပြုလုပ်နိုင်သောကြောင့်) ပိုမို အဆင်ပြေစေပါသည်။ Syntax မှာ `scp path/to/local_file remote_host:path/to/remote_file` ဖြစ်ပါသည်။
- **`rsync`** : `rsync` သည် local နှင့် remote တွင် တူညီသော ဖိုင်များကို ထောက်လှမ်းသိရှိပြီး ၎င်းတို့ကို ထပ်မံ ကူးယူခြင်းမှ တားဆီးပေးခြင်းဖြင့် `scp` ထက် ပိုမို ကောင်းမွန်အောင် ပြုလုပ်ထားပါသည်။ ယင်းသည် symlink များ၊ permission များနှင့် ပတ်သက်၍ ပိုမို အသေးစိတ် စီမံခန့်ခွဲနိုင်စေပြီး ယခင်က ပြတ်တောက်သွားခဲ့သော ကူးယူမှုကို ပြန်လည် စတင်နိုင်သည့် `--partial` flag ကဲ့သို့သော အပိုဆောင်း feature များလည်း ပါဝင်ပါသည်။ `rsync` တွင် `scp` နှင့် ဆင်တူသော syntax ရှိပါသည်။

Process များကို Background တွင် လည်ပတ်စေခြင်း

Default အားဖြင့် ssh ချိတ်ဆက်မှု ပြတ်တောက်သွားသည့်အခါ parent shell ၏ child process များသည် ယင်းနှင့်အတူ သေဆုံးသွားကြပါသည်။ အခြား ရွေးချယ်စရာ နည်းလမ်းအချို့ ရှိကြပါသည် -

- **`nohup`** : `nohup` tool သည် terminal သေဆုံးသွားသည့်အခါတွင်ပင် process တစ်ခု ဆက်လက် အသက်ရှင်နေစေရန် ထိရောက်စွာ ဆောင်ရွက်ပေးပါသည်။ ဤအရာကို တစ်ခါတစ်ရံတွင် `&` နှင့် `disown` တို့ဖြင့် ပြုလုပ်နိုင်သော်လည်း `nohup` သည် ပိုမိုကောင်းမွန်သော default တစ်ခု ဖြစ်ပါသည်။ အသေးစိတ်ကို [ဒီနေရာတွင်](https://unix.stackexchange.com/questions/3886/difference-between-nohup-disown-and) ရှာဖွေနိုင်ပါသည်။

- `tmux` ၊ `screen` : `nohup` သည် process ကို background သို့ ထိရောက်စွာ ပို့ဆောင်ပေးသော်လည်း interactive shell session များအတွက်တော့ အဆင်မပြေလှပါ။ ထိုသို့သော အခြေအနေမျိုးတွင် `screen` သို့မဟုတ် `tmux` ကဲ့သို့သော terminal multiplexer တစ်ခုကို အသုံးပြုခြင်းသည် သက်ဆိုင်ရာ shell များကို လွယ်ကူစွာ detach နှင့် reattach ပြုလုပ်နိုင်သောကြောင့် အဆင်ပြေသည့် ရွေးချယ်မှု ဖြစ်ပါသည်။

နောက်ဆုံးအနေဖြင့် အကယ်၍ သင်သည် ပရိုဂရမ်တစ်ခုကို disown ပြုလုပ်ခဲ့ပြီး ယင်းကို လက်ရှိ terminal ထံ ပြန်လည် reattach ပြုလုပ်လိုပါက **reptyr** (https://github.com/nelhage/reptyr) ကို လေ့လာကြည့်နိုင်ပါသည်။ `reptyr PID` သည် ID PID ပါရှိသော process ကို ရယူပြီး သင်၏ လက်ရှိ terminal သို့ တွဲဆက် (attach) ပေးမည် ဖြစ်သည်။

Port Forwarding

အခြေအနေ အများအပြားတွင် သင်သည် စက်အတွင်းရှိ port များကို နားထောင်ခြင်း (listen လုပ်ခြင်း) ဖြင့် အလုပ်လုပ်သော ဆော့ဖ်ဝဲလ်များနှင့် ကြုံတွေ့ရမည် ဖြစ်သည်။ ဤသို့ သင်၏ local စက်တွင် ဖြစ်ပျက်သည့် အခါ `localhost:PORT` သို့မဟုတ် `127.0.0.1:PORT` ကို ရိုးရှင်းစွာ ပြုလုပ်နိုင်သော်လည်း၊ ကွန်ရက်/အင်တာနက်မှတစ်ဆင့် ယင်း၏ port များကို တိုက်ရိုက် ရယူသုံးစွဲနိုင်ခြင်း မရှိသော remote server တစ်ခုနှင့် ကြုံလျှင် မည်သို့ ပြုလုပ်မည်နည်း။ ဤသည်ကို port forwarding ဟု ခေါ်ဆိုပြီး ယင်းတွင် ပုံစံနှစ်မျိုး ပါဝင်ပါသည်- Local Port Forwarding နှင့် Remote Port Forwarding (အသေးစိတ်အတွက် ပုံများကို ကြည့်ပါ၊ ပုံများ၏ credit မှာ **ဤ SO post** (https://unix.stackexchange.com/questions/115897/whats-ssh-port-forwarding-and-whats-the-difference-between-ssh-local-and-remot) မှ ဖြစ်ပါသည်)။

Local Port Forwarding ![Local Port Forwarding](https://i.stack.imgur.com/a28N8.png) (https://i.stack.imgur.com/a28N8.png)

Remote Port Forwarding ![Remote Port Forwarding](https://i.stack.imgur.com/4iK3b.png) (https://i.stack.imgur.com/4iK3b.png)

အတွေ့ရအများဆုံး အခြေအနေမှာ local port forwarding ဖြစ်ပြီး၊ remote စက်အတွင်းရှိ service တစ်ခုက port တစ်ခုတွင် နားထောင်နေကာ သင်က remote port ထံ ပေးပို့ရန် (forward ပြုလုပ်ရန်) အတွက် သင်၏ local စက်ရှိ port တစ်ခုကို ချိတ်ဆက်လိုခြင်း ဖြစ်ပါသည်။ ဥပမာအားဖြင့် remote server တွင် port `8888` ကို နားထောင်နေသည့် `jupyter notebook` ကို မောင်းနှင်လိုက်သည် ဆိုပါစို့။ ထို့ကြောင့် ယင်းကို local port `9999` သို့ forward ပြုလုပ်ရန်အတွက် `ssh -L 9999:localhost:8888 foobar@remote_server` ကို ပြုလုပ်မည်ဖြစ်ပြီး၊ ထို့နောက် မိမိတို့ local စက်တွင် `localhost:9999` သို့ ဝင်ရောက် ကြည့်ရှုမည် ဖြစ်သည်။

Graphics Forwarding

တစ်ခါတစ်ရံတွင် server ၌ GUI အခြေပြု ပရိုဂရမ်တစ်ခုကို မောင်းနှင်လိုသောကြောင့် port များကို forward ပြုလုပ်ရုံဖြင့် မလုံလောက်ပါ။ Desktop Environment တစ်ခုလုံးကို ပေးပို့ပေးသည့် Remote Desktop Software များကို အမြဲတမ်း အသုံးပြုနိုင်ပါသည် (ဥပမာ RealVNC, Teamviewer စသည့် ရွေးချယ်စရာများ)။ သို့သော်လည်း သီးခြား GUI tool တစ်ခုတည်းအတွက်မူ SSH သည် ကောင်းမွန်သော အခြားရွေးချယ်စရာတစ်ခုကို ပေးစွမ်းထားပါသည်- Graphics Forwarding ဖြစ်ပါသည်။

`-X` flag ကို အသုံးပြုခြင်းဖြင့် X11 ကို forward ပြုလုပ်ရန် SSH အား ညွှန်ကြားပါသည်။

ယုံကြည်ရသော (trusted) X11 forwarding အတွက် `-Y` flag ကို အသုံးပြုနိုင်ပါသည်။

နောက်ဆုံး သတိပြုရန်အချက်မှာ ဤသည် အလုပ်လုပ်ရန်အတွက် server ပေါ်ရှိ `sshd_config` တွင် အောက်ပါ option များ ရှိနေရမည် ဖြစ်ပါသည် -

```
X11Forwarding yes
X11DisplayOffset 10
```

Roaming

Remote server တစ်ခုသို့ ချိတ်ဆက်သည့်အခါ ကြုံတွေ့ရလေ့ရှိသော အခက်အခဲတစ်ခုမှာ မိမိ၏ ကွန်ပျူတာကို ပိတ်လိုက်ခြင်း/sleep လုပ်လိုက်ခြင်း သို့မဟုတ် ကွန်ရက် ပြောင်းလဲလိုက်ခြင်းတို့ကြောင့် ချိတ်ဆက်မှု ပြတ်တောက်သွားခြင်း ဖြစ်ပါသည်။ ထို့အပြင် အကယ်၍ ချိတ်ဆက်မှုတွင် သိသာသော lag ရှိနေပါက ssh ကို အသုံးပြုခြင်းသည် အတော်လေး စိတ်ပျက်စရာ ကောင်းလာနိုင်ပါသည်။ Mobile shell ဖြစ်သော **Mosh** (https://mosh.org/) သည် ssh ကို ပို့မိုကောင်းမွန်အောင် ပြုလုပ်ထားပြီး roaming ချိတ်ဆက်မှုများ၊ ပြတ်တောင်းပြတ်တောင်း ချိတ်ဆက်နိုင်မှုများကို ခွင့်ပြုပေးသည့်အပြင် ဉာဏ်ရည်ထက်မြက်သော local echo ကိုလည်း ပေးစွမ်းနိုင်ပါသည်။

Mosh ကို အသုံးများသော distribution များနှင့် package manager များ အားလုံးတွင် ရယူနိုင်ပါသည်။ Mosh သည် server အတွင်း၌ ssh server တစ်ခု အလုပ်လုပ်နေရန် လိုအပ်ပါသည်။ Mosh ကို ထည့်သွင်းရန်အတွက် သင်သည် superuser ဖြစ်ရန် မလိုအပ်သော်လည်း server တွင် port 60000 မှ 60010 အထိ ပွင့်နေရန် လိုအပ်ပါသည်။ (၎င်းတို့သည် privileged range အတွင်း မရှိသောကြောင့် ပုံမှန်အားဖြင့် ပွင့်နေလေ့ ရှိပါသည်)။

`mosh` ၏ အားနည်းချက်တစ်ခုမှာ ယင်းသည် roaming port/graphics forwarding ကို ထောက်ပံ့မပေးခြင်း ဖြစ်သောကြောင့် အကယ်၍ သင်သည် ၎င်းတို့ကို မကြာခဏ သုံးစွဲပါက `mosh` သည် များစွာ အကူအညီ ဖြစ်မည် မဟုတ်ပါ။

SSH Configuration

Client ပိုင်း

ကျွန်ုပ်တို့သည် ပေးပို့နိုင်သည့် argument အများအပြားကို ဆွေးနွေးခဲ့ကြပြီး ဖြစ်သည်။ စိတ်ဝင်စားစရာ အခြားရွေးချယ်စရာတစ်ခုမှာ `alias my_server="ssh -X -i ~/.ssh/id_rsa -L 9999:localhost:8888 foobar@remote_server"` ကဲ့သို့သော shell alias များကို ဖန်တီးခြင်း ဖြစ်သော်လည်း၊ `~/.ssh/config` ကို အသုံးပြုခြင်းဟူသော ပိုမိုကောင်းမွန်သည့် အခြားရွေးချယ်စရာတစ်ခု ရှိပါသည်။

```
Host vm
    User foobar
    HostName 172.16.174.141
    Port 22
    IdentityFile ~/.ssh/id_rsa
    RemoteForward 9999 localhost:8888

# Configs can also take wildcards
Host *.mit.edu
    User foobaz
```

Alias များထက် `~/.ssh/config` ဖိုင်ကို အသုံးပြုခြင်း၏ အပိုဆောင်း အားသာချက်တစ်ခုမှာ `scp`၊ `rsync` ၊ `mosh` စသည့် အခြားသော ပရိုဂရမ်များသည်လည်း ယင်းကို ဖတ်ရှုနိုင်ပြီး အပြင်အဆင်များကို သက်ဆိုင်ရာ flag များအဖြစ် ပြောင်းလဲပေးနိုင်ခြင်း ဖြစ်ပါသည်။

`~/.ssh/config` ဖိုင်ကို dotfile တစ်ခုအဖြစ် သတ်မှတ်နိုင်ပြီး၊ ယေဘုယျအားဖြင့် ယင်းကို သင်၏ အခြားသော dotfile များနှင့်အတူ ထည့်သွင်းထားခြင်းသည် အဆင်ပြေပါသည်။ သို့သော်လည်း အကယ်၍ သင်သည် ယင်းကို အများပြည်သူသို့ ထုတ်ဖော်ပြသလိုက်ပါက (public ပြုလုပ်လိုက်ပါက) အင်တာနက်ပေါ်ရှိ ပြင်ပသူများထံ သင် ပေးအပ်လိုက်နိုင်သည့် အချက်အလက်များကို စဉ်းစားကြည့်ပါ- သင်၏ server များ၏ လိပ်စာများ၊ သင် အသုံးပြုနေသော user များ၊ ပွင့်နေသော port များ စသည်တို့ ဖြစ်ပါသည်။ ဤသည်မှာ အချို့သော တိုက်ခိုက်မှု အမျိုးအစားများကို လွယ်ကူချောမွေ့သွားစေနိုင်သဖြင့် သင်၏ SSH configuration ကို မျှဝေရာတွင် သေချာ စဉ်းစားဆင်ခြင်ပါ။

သတိပေးချက်- သင်၏ RSA key များကို (`~/.ssh/id_rsa*`) public repository တွင် မည်သည့်အခါမျှ ထည့်သွင်းခြင်း မပြုပါနှင့်!

Server ပိုင်း

Server ပိုင်း configuration ကို ပုံမှန်အားဖြင့် `/etc/ssh/sshd_config` တွင် သတ်မှတ်လေ့ရှိပါသည်။ ဤနေရာတွင် သင်သည် password ဖြင့် အတည်ပြုစစ်ဆေးခြင်းကို ပိတ်ထားခြင်း၊ ssh port များကို ပြောင်းလဲခြင်း၊ X11 forwarding ကို ဖွင့်လှစ်ခြင်း စသည့် ပြောင်းလဲမှုများကို ပြုလုပ်နိုင်ပါသည်။ Config အပြင်အဆင်များကို user တစ်ဦးချင်းစီအလိုက် သတ်မှတ်ပေးနိုင်ပါသည်။

Remote Filesystem (အဝေးထိန်း ဖိုင်စနစ်)

တစ်ခါတစ်ရံတွင် remote folder တစ်ခုကို mount ပြုလုပ်ထားခြင်းက အဆင်ပြေစေပါသည်။ **sshfs** (https://github.com/libfuse/sshfs) သည် remote server ပေါ်ရှိ folder တစ်ခုကို local တွင် mount ပြုလုပ်ပေးနိုင်ပြီး၊ ထို့နောက် သင်သည် local editor တစ်ခုကို အသုံးပြုနိုင်မည် ဖြစ်သည်။

လေ့ကျင့်ခန်းများ

1. SSH အလုပ်လုပ်ရန်အတွက် host စက်တွင် SSH server တစ်ခု runနေရန် လိုအပ်ပါသည်။ ကျန်ရှိသော လေ့ကျင့်ခန်းများကို လုပ်ဆောင်နိုင်ရန်အတွက် virtual machine တစ်ခုအတွင်း၌ SSH server တစ်ခု (ဥပမာ OpenSSH) ထည့်သွင်းပါ။ စက်၏ IP မည်မျှဖြစ်သည်ကို သိရှိနိုင်ရန် `ip addr` command ကို runပြီး `inet` နေရာကို ရှာဖွေပါ (loopback interface နှင့် သက်ဆိုင်သော `127.0.0.1` စာရင်းကို ကျော်ပါ)။

2. `~/.ssh/` သို့ သွားရောက်ပြီး ထိုနေရာတွင် SSH key အတွဲတစ်ခု ရှိမရှိ စစ်ဆေးပါ။ မရှိပါက `ssh-keygen -t rsa -b 4096` ဖြင့် ထုတ်လုပ်ပါ။ သင့်အနေဖြင့် password တစ်ခုကို အသုံးပြုပြီး `ssh-agent` ကို သုံးစွဲရန် အကြံပြုပါသည်၊ အသေးစိတ် အချက်အလက်များကို **ဒီနေရာတွင်** (https://www.ssh.com/ssh/agent) ကြည့်ပါ။

3. Key ကို မိမိ၏ virtual machine ထံ ကူးယူရန် `ssh-copy-id` ကို အသုံးပြုပါ။ Password မလိုဘဲ ssh ဝင်ရောက်နိုင်မနိုင် စမ်းသပ်ပါ။ ထို့နောက် server ပေါ်ရှိ သင်၏ `sshd_config` ကို ပြင်ဆင်ပြီး `PasswordAuthentication` ၏ တန်ဖိုးကို ပြောင်းလဲကာ password ဖြင့် အတည်ပြုစစ်ဆေးခြင်းကို ပိတ်ပါ။ `PermitRootLogin` ၏ တန်ဖိုးကို ပြောင်းလဲခြင်းဖြင့် root login ဝင်ရောက်ခြင်းကို ပိတ်ပါ။

4. SSH port ကို ပြောင်းလဲရန် server ရှိ `sshd_config` ကို ပြင်ဆင်ပြီး ssh ဝင်ရောက်နိုင်ဆဲ ဟုတ်မဟုတ် စစ်ဆေးပါ။ အကယ်၍ သင့်တွင် အများပြည်သူ ဝင်ရောက်နိုင်သော (public facing) server တစ်ခု ရှိပါက default မဟုတ်သော port တစ်ခုနှင့် key သာ သုံးသည့် login တို့သည် မသမာသော တိုက်ခိုက်မှု ပမာဏ အမြောက်အများကို ဟန့်တားပေးနိုင်မည် ဖြစ်သည်။

5. သင်၏ server/VM တွင် mosh ကို ထည့်သွင်းပါ၊ ချိတ်ဆက်မှုတစ်ခု ပြုလုပ်ပြီးနောက် server/VM ၏ network adapter ကို ဖြုတ်လိုက်ပါ။ Mosh သည် ယင်းမှ အဆင်ပြေစွာ ပြန်လည် စတင်နိုင်ပါသလား (recover ဖြစ်ပါသလား)။

6. Local port forwarding ၏ အခြားသော အသုံးပြုမှုတစ်ခုမှာ သီးခြား host တစ်ခုကို server သို့ tunnel ဖောက်၍ ဆက်သွယ်ခြင်း ဖြစ်သည်။ အကယ်၍ သင်၏ ကွန်ရက်သည် ဥပမာ `reddit.com` ကဲ့သို့သော ဝဘ်ဆိုက်အချို့ကို စစ်ထုတ်ထားပါက ယင်းကို server မှတစ်ဆင့် အောက်ပါအတိုင်း tunnel ဖောက်၍ ဝင်ရောက်နိုင်သည် -

```
- `ssh remote_server -L 80:reddit.com:80` ကို runပါ
- `/etc/hosts` တွင် `reddit.com` နှင့် `www.reddit.com` တို့ကို `127.0.0.1` ဟု သတ်မှတ်ပ
ါ
- ထိုဝဘ်ဆိုက်သို့ server မှတစ်ဆင့် ဝင်ရောက်နေခြင်း ဟုတ်မဟုတ် စစ်ဆေးပါ
- သိသာထင်ရှားမှု မရှိပါက သင်၏ host public IP အပေါ် မူတည်၍ ပြောင်းလဲမည်ဖြစ်သော <a href
="https://ipinfo.io/" class="ext-link">ipinfo.io</a> <span class="ext-url-prin
t">(https://ipinfo.io/)</span> ကဲ့သို့သော ဝဘ်ဆိုက်တစ်ခုကို အသုံးပြုပါ
```

1. Background port forwarding ကို အပိုဆောင်း flag အနည်းငယ်ဖြင့် လွယ်ကူစွာ ပြုလုပ်နိုင်ပါသည်။ `ssh` တွင် `-N` နှင့် `-f` flag များက မည်သည့်အရာ ပြုလုပ်သည်ကို စုံစမ်းလေ့လာပြီး `ssh -N -f -L 9999:localhost:8888 foobar@remote_server` ကဲ့သို့သော command တစ်ခုက မည်သည့်အရာ ပြုလုပ်သည်ကို အဖြေရှာပါ။

ကိုးကားချက်များ

- **SSH Hacks** (https://matt.might.net/articles/ssh-hacks/)
- **Secure Secure Shell** (https://stribika.github.io/2015/01/04/secure-secure-shell.html)

🔗 External Links & References (ပြင်ပ လင့်ခ်များနှင့် ကိုးကားချက်များ)

Resource / Description	URL	Micro QR
1. ဤနေရာတွင်	https://help.github.com/articles/connecting-to-github-with-ssh/	
2. ဒီနေရာတွင်	https://unix.stackexchange.com/questions/3886/difference-between-nohup-disown-and	
3. reptyr	https://github.com/nelhage/reptyr	
4. ဤ SO post	https://unix.stackexchange.com/questions/115897/whats-ssh-port-forwarding-and-whats-the-difference-between-ssh-local-and-remot	
5. Local Port Forwarding	https://i.stack.imgur.com/a28N8.png	
6. Remote Port Forwarding	https://i.stack.imgur.com/4iK3b.png	
7. Mosh	https://mosh.org/	
8. sshfs	https://github.com/libfuse/sshfs	
9. ဒီနေရာတွင်	https://www.ssh.com/ssh/agent	
10. ipinfo.io	https://ipinfo.io/	
11. SSH Hacks	https://matt.might.net/articles/ssh-hacks/	
12. Secure Secure Shell	https://stribika.github.io/2015/01/04/secure-secure-shell.html	

Security and Privacy

▶ LECTURE VIDEO REFERENCE

Security and Privacy

Scan the QR code below using your mobile camera or click the link to watch this full lecture online:

Direct Online Link:

https://www.youtube.com/watch?v=OBx_c-i-M8s

ကမ္ဘာကြီးသည် ကြောက်စရာကောင်းသော နေရာတစ်ခုဖြစ်ပြီး လူတိုင်းက သင့်ကို ဒုက္ခပေးရန် စောင့်ဆိုင်းနေကြသည်။

ကဲ၊ အမှန်တကယ်တော့ ထိုသို့မဟုတ်ကောင်း မဟုတ်နိုင်သော်လည်း၊ ယင်းက သင့်လျှို့ဝှက်ချက်အားလုံးကို ပေါပေါထင်ထင် ထုတ်ပြသင့်သည်ဟု မဆိုလိုပါ။ လုံခြုံရေး (နှင့် သီးသန့်လုံခြုံမှု) ဆိုသည်မှာ ယေဘုယျအားဖြင့် တိုက်ခိုက်သူများအတွက် ခက်ခဲအောင် အတားအဆီး မြှင့်တင်ပေးခြင်း ဖြစ်သည်။ သင့်တွင် မည်

သည့် ခြိမ်းခြောက်မှု မူဘောင် (threat model) ရှိသည်ကို ရှာဖွေပြီး ထိုအပေါ် အခြေခံ၍ လုံခြုံရေး နည်းလမ်းများကို ဒီဇိုင်းထုတ်ပါ! သင့် ခြိမ်းခြောက်မှု မူဘောင်သည် NSA သို့မဟုတ် Mossad ဖြစ်နေပါက သင့်အတွက် ဒုက္ခလှလှ ရောက်ရပေလိမ့်မည်။

သင့်နည်းပညာဆိုင်ရာ ကိုယ်ရည်ကိုယ်သွေး (technical persona) ကို ပိုမိုလုံခြုံစေရန် နည်းလမ်း *များစွာ* ရှိပါသည်။ ဤနေရာတွင် အထွေထွေ အဆင့်မြင့် အရာအများအပြားကို ဆွေးနွေးသွားမည် ဖြစ်သော်လည်း၊ ဤသည်မှာ စဉ်ဆက်မပြတ် လုပ်ဆောင်ရမည့် လုပ်ငန်းစဉ် တစ်ခုဖြစ်ပြီး မိမိကိုယ်ကို လေ့လာသင်ယူခြင်းသည် သင်ပြုလုပ်နိုင်သည့် အကောင်းဆုံး အရာများထဲမှ တစ်ခုဖြစ်သည်။ ထို့ကြောင့် -

သင့်တော်သော လူများကို Follow လုပ်ပါ (Follow the Right People)

သင့်လုံခြုံရေးဆိုင်ရာ ဗဟုသုတများကို မြှင့်တင်ရန် အကောင်းဆုံး နည်းလမ်းတစ်ခုမှာ လုံခြုံရေးအကြောင်း ပြောဆိုဆွေးနွေးလေ့ရှိသော အခြားသူများကို Follow လုပ်ခြင်း ဖြစ်သည်။ အကြံပြုချက် အချို့မှာ -

- **@TroyHunt** (https://twitter.com/TroyHunt)
- **@SwiftOnSecurity** (https://twitter.com/SwiftOnSecurity)
- **@taviso** (https://twitter.com/taviso)
- **@thegrugq** (https://twitter.com/thegrugq)
- **@tqbf** (https://twitter.com/tqbf)
- **@mattblaze** (https://twitter.com/mattblaze)
- **@moxie** (https://twitter.com/moxie)

ပိုမိုသိရှိလိုပါက **ဤစာရင်း (this list)** (https://heimdalsecurity.com/blog/best-twitter-cybersec-accounts/) ကိုလည်း ကြည့်ရှုပါ။

ယေဘုယျ လုံခြုံရေး အကြံပြုချက်များ (General Security Advice)

Tech Solidarity တွင် လက်တွေ့ကျသော အကြံပြုချက်များစွာ ပါဝင်ပြီး အတော်အတန် ခေတ်မီဆန်းသစ်သော **သတင်းထောက်များအတွက် ပြုလုပ်ရန်နှင့် ရှောင်ကြဉ်ရန် အချက်များ (do's and don'ts for journalists)** (https://web.archive.org/web/20221123204419/https://techsolidarity.org/resources/basic_security.htm) စာရင်းကောင်းတစ်ခု ရှိပါသည်။ **@thegrugq** (https://medium.com/@thegrugq) တွင်လည်း ဖတ်ရှုသင့်သော **ခရီးသွားလာရေး လုံခြုံရေး အကြံပြုချက် (travel security advice)** (https://medium.com/@thegrugq/stop-fabricating-travel-security-advice-35259bf0e869) ဘလော့ဂ်ဆောင်းပါးကောင်း တစ်ခုရှိသည်။ ထိုရင်းမြစ်များမှ အကြံပြုချက်အများအပြားနှင့် အခြားအချက်များကို ဤနေရာတွင် ထပ်မံဖော်ပြပေးပါမည်။ ထို့အပြင် **USB သည့်**

ကြောက်စရာကောင်းသောကြောင့် (https://www.bleepingcomputer.com/news/security/heres-a-list-of-29-different-types-of-usb-attacks/) **USB data blocker** (https://www.amazon.com/dp/B00QRRZ2QM/) တစ်ခု ဝယ်ယူထားပါ။

အထောက်အထား စိစစ်ခြင်း (Authentication)

သင် မပြုလုပ်ရသေးပါက ပထမဆုံး ပြုလုပ်သင့်သည့် အရာမှာ password manager တစ်ခု ဒေါင်းလုဒ်လုပ်ခြင်း ဖြစ်သည်။ သုံးစွဲရန် ကောင်းမွန်သော အချို့မှာ -

- **1password** (https://1password.com/)
- **KeePass** (https://keepass.info/)
- **BitWarden** (https://bitwarden.com/)
- **pass** (https://git.zx2c4.com/password-store/about/)

အကယ်၍ သင်သည် အလွန်အမင်း စိုးရိမ်တတ်ပါက စာဗာတွင် Plain-text အဖြစ် သိမ်းဆည်းခြင်း မဟုတ်ဘဲ သင့်ကွန်ပျူတာပေါ်တွင် လျှို့ဝှက်ချက်များကို Local အလိုက် Encrypt ပြုလုပ်ပေးသည့် အရာကို အသုံးပြုပါ။ သင် အရေးစိုက်သော ဝဘ်ဆိုက်အားလုံးအတွက် စကားဝှက်များ ထုတ်ယူရန် ယင်းကို ချက်ချင်း အသုံးပြုပါ။ ထို့နောက် Two-factor authentication (၂ ဆင့် အထောက်အထား စိစစ်ခြင်း) ကို ဖွင့်ပါ၊ အထူးသဖြင့် **FIDO/U2F** (https://fidoalliance.org/) dongle ဖြင့် ပြုလုပ်ခြင်းက အကောင်းဆုံး ဖြစ်သည် (ဥပမာ **YubiKey** (https://www.yubico.com/quiz/) ကဲ့သို့သော အရာဖြစ်ပြီး **ကျောင်းသားများအတွက် ၂၀% လျှော့ဈေး** (https://www.yubico.com/why-yubico/for-education/) ရှိသည်)။ TOTP (Google Authenticator သို့မဟုတ် Duo ကဲ့သို့သော) သည်လည်း လိုအပ်ပါက သုံးနိုင်သော်လည်း **Phishing ကို ကာကွယ်မပေးနိုင်ပါ** (https://twitter.com/taviso/status/1082015009348104192)။ သင့် ခြိမ်းခြောက်မှု မူဘောင်တွင် လမ်းခုလတ်မှ သင့် စကားဝှက်ကို ကြိုရာလူ ရယူသွားခြင်း တစ်ခုတည်းသာ ပါဝင်ပါက လွဲ၍ SMS သည် အသုံးမဝင်သလောက် ဖြစ်သည်။

ထို့အပြင် စက္ကူသော့များ (paper keys) အကြောင်း မှတ်ချက်တစ်ခု။ ဝန်ဆောင်မှုများသည် သင်၏ အစစ်အမှန် ဒုတိယ အချက်အလက် ဆုံးရှုံးသွားပါက သုံးနိုင်သည့် “backup key” တစ်ခုကို ပေးလေ့ရှိကြသည် (စကားမစပ်၊ အရန် dongle တစ်ခုကို အမြဲတမ်း ဘေးကင်းသော နေရာတွင် သိမ်းဆည်းထားပါ!)။ ယင်းတို့ကို သင့် password manager တွင် ထည့်ထားနိုင်သော်လည်း တစ်စုံတစ်ယောက်က သင့် password manager ကို ရယူသွားပါက သင့်အတွက် အကုန်လုံး အန္တရာယ်ရှိသွားမည် ဖြစ်သည် (သို့သော် ထို ခြိမ်းခြောက်မှု မူဘောင်ကို သင် လက်ခံနိုင်ပေလိမ့်မည်)။ အကယ်၍ သင်သည် အမှန်တကယ် စိုးရိမ်တတ်ပါက ဤ စက္ကူသော့များကို ပရင့်ထုတ်ပါ၊ ဒီဂျစ်တယ်စနစ်ဖြင့် ဘယ်တော့မှ မသိမ်းဆည်းပါနှင့်၊ ထို့နောက် အပြင်ကမ္ဘာရှိ လုံခြုံသော မီးခံသေတ္တာထဲတွင် ထည့်ထားပါ။

သီးသန့် ဆက်သွယ်ပြောဆိုခြင်း (Private Communication)

Signal (https://www.signal.org/) ကို အသုံးပြုပါ (**တပ်ဆင်နည်း ညွှန်ကြားချက်များ** (https://medium.com/@mshelton/signal-for-beginners-c6b44f76a1f0)။ **Wire** (https://wire.com/en/) ကိုလည်း **သုံးနိုင်ပါသည်** (https://www.securemessagingapps.com/)၊ WhatsApp သည်လည်း အဆင်ပြေပါသည်၊ **Telegram ကို မသုံးပါနှင့်** (https://twitter.com/bascule/status/897187286554628096))။ Desktop messenger များသည် အလွန် အားနည်းပါသည် (အဓိကအားဖြင့် ကြီးမားသော ယုံကြည်မှု အစုအဝေး ဖြစ်သည့် Electron ကို အားကိုးလေ့ ရှိသောကြောင့် ဖြစ်သည်)။

E-mail သည် PGP လက်မှတ်ထိုးထားလျှင်ပင် အထူးသဖြင့် ပြဿနာများစွာ ရှိသည်။ ယင်းသည် ယေဘုယျ အားဖြင့် Forward-secure မဖြစ်သလို key-distribution ပြဿနာမှာလည်း အလွန်ဆိုးရွားပါသည်။ **keybase.io** (https://keybase.io/) က ကူညီပေးနိုင်ပြီး အခြား အကြောင်းပြချက်များစွာအတွက် အသုံးဝင်ပါသည်။ ထို့အပြင် PGP key များကို ယေဘုယျအားဖြင့် Desktop ကွန်ပျူတာများပေါ်တွင် ကိုင်တွယ်လေ့ရှိပြီး ယင်းသည် လုံခြုံရေး အနည်းဆုံး ကွန်ပျူတာ ပတ်ဝန်းကျင်များထဲမှ တစ်ခုဖြစ်သည်။ ဆက်စပ်၍ Chromebook တစ်ခု ဝယ်ယူရန် စဉ်းစားပါ၊ သို့မဟုတ် ကီးဘုတ်ပါသော Tablet ပေါ်တွင်သာ အလုပ်လုပ်ပါ။

ဖိုင် လုံခြုံရေး (File Security)

ဖိုင် လုံခြုံရေးသည် ခက်ခဲပြီး အဆင့်များစွာတွင် လုပ်ဆောင်သည်။ သင်သည် မည်သည့်အရာကို ကာကွယ်ရန် ကြိုးစားနေသနည်း။

- **အော့ဖ်လိုင်း တိုက်ခိုက်မှုများ (အော့ဖ်ဖြစ်နေချိန်တွင် သင့်လက်ပ်တော့ကို တစ်စုံတစ်ယောက်က ခိုးယူသွားခြင်း) - Full disk encryption ကို ဖွင့်ပါ။ (Linux တွင် cryptsetup + LUKS** (https://wiki.archlinux.org/index.php/Dm-crypt/Encrypting_a_non-root_file_system)၊ Windows တွင် **BitLocker** (https://fossbytes.com/enable-full-disk-encryption-windows-10/)၊ macOS တွင် **FileVault** (https://support.apple.com/en-us/HT204837))။ တိုက်ခိုက်သူက သင့်ကိုပါ ဖမ်းဆီးရရှိထားပြီး သင့်လျှို့ဝှက်ချက်များကို တကယ်လိုချင်နေပါက ဤသည်က ကူညီနိုင်မည် မဟုတ်ကြောင်း သတိပြုပါ)။
- အွန်လိုင်း တိုက်ခိုက်မှုများ (သင့်လက်ပ်တော့ ပွင့်နေချိန်တွင် တစ်စုံတစ်ယောက်က ရရှိသွားခြင်း) - ဖိုင် Encrypt ပြုလုပ်ခြင်းကို အသုံးပြုပါ။ ယင်းအတွက် အဓိက နည်းလမ်း နှစ်ခုရှိသည် -

```
- Encrypted filesystems - အဆင့်ဆင့် ပြုလုပ်ထားသော Filesystem encryption စောဖ်ဝဲလ်သ
ည် Encrypted block device များအစား ဖိုင်များကို တစ်ခုချင်းစီ Encrypt ပြုလုပ်သည်။ Decryptio
n key ပေးသွင်းခြင်းဖြင့် ဤ Filesystem များကို "mount" ပြုလုပ်နိုင်ပြီး အတွင်းရှိ ဖိုင်များကို လွတ်လပ်
စွာ ဝင်ရောက် ကြည့်ရှုနိုင်သည်။ "unmount" ပြုလုပ်လိုက်သည့်အခါ ထိုဖိုင်များအားလုံးကို ရရှိနိုင်တော့မည် မ
ဟုတ်ပါ။ ခေတ်မီ နည်းလမ်းများတွင် <a href="https://github.com/rfjakob/gocryptfs" class
="ext-link">gocryptfs</a> <span class="ext-url-print">(https://github.com/rfjak
ob/gocryptfs)</span> နှင့် <a href="https://www.ecryptfs.org/" class="ext-link">eC
ryptFS</a> <span class="ext-url-print">(https://www.ecryptfs.org/)</span> တို့ ပါဝ
င်သည်။ ပိုမိုအသေးစိတ်ကျသော နှိုင်းယှဉ်ချက်များကို <a href="https://nuetzlich.net/gocryptf
s/comparison/" class="ext-link">ဤနေရာ</a> <span class="ext-url-print">(http
s://nuetzlich.net/gocryptfs/comparison/)</span> နှင့် <a href="https://wiki.archli
nux.org/index.php/disk_encryption#Comparison_table" class="ext-link">ဤနေရာ</a>
<span class="ext-url-print">(https://wiki.archlinux.org/index.php/disk_encrypti
on#Comparison_table)</span> တွင် ကြည့်ရှုနိုင်ပါသည်။
- Encrypted files - သီးခြား ဖိုင်များကို Symmetric encryption (`gpg -c` ကိုကြည့်ပါ) နှင့်
လျှို့ဝှက်သော Key ဖြင့် Encrypt ပြုလုပ်ပါ။ သို့မဟုတ် `pass` ကဲ့သို့ပင် သင့် Private key ဖြင့်သာ န
ောက်မှ ပြန်လည်ဖတ်ရှုနိုင်စေရန် Key ကို သင့် Public key ဖြင့် Encrypt ပြုလုပ်ပါ။ တိကျသော Encr
yption ဆက်တင်များသည် အလွန်အရေးကြီးပါသည်!
```

- [ယုတ္တိတန်စွာ ငြင်းဆိုနိုင်မှု (Plausible deniability)](https://en.wikipedia.org/wiki/Plausible_deniability) (အရာရှိမင်း ဘာပြဿနာများ ရှိလို့လဲ) - ပုံမှန်အားဖြင့် Performance ပိုမိုနိမ့်ကျပြီး ဒေတာ ဆုံးရှုံးရန် ပိုမိုလွယ်ကူသည်။ ယင်းက [deniable encryption](https://en.wikipedia.org/wiki/Deniable_encryption) ပေးဆောင်ကြောင်း အမှန်တကယ် သက်သေပြရန် ခက်ခဲပါသည်! [ဤနေရာရှိ ဆွေးနွေးချက်](https://security.stackexchange.com/questions/135846/is-plausible-deniability-actually-feasible-for-encrypted-volumes-disks) ကို ကြည့်ပါ၊ ထို့နောက် [VeraCrypt](https://www.veracrypt.fr/en/Home.html) (ယခင် TrueCrypt ၏ ထိန်းသိမ်းထားသော Fork ဖြစ်သည်) ကို စမ်းသပ်ကြည့်လိုခြင်း ရှိမရှိ စဉ်းစားပါ။
- Encrypted backups - [Tarsnap](https://www.tarsnap.com/) သို့မဟုတ် [Borgbase](https://www.borgbase.com/) ကို အသုံးပြုပါ

```
- တိုက်ခိုက်သူက သင့်လက်ပ်တော့ကို ရရှိသွားပါက သင့်အရန်သိမ်းဆည်းမှု (backups) များကို ဖျက်ဆီးနိုင်မ
နိုင် စဉ်းစားပါ!
```

အင်တာနက် လုံခြုံရေး နှင့် သီးသန့်လုံခြုံမှု (Internet Security & Privacy)

အင်တာနက်သည် အလွန် ကြောက်စရာကောင်းသော နေရာတစ်ခုဖြစ်သည်။ Open WiFi ကွန်ရက်များသည် [ကြောက်စရာကောင်းပါသည်](https://www.troyhunt.com/the-beginners-guide-to-breaking-website/) [ကြောက်စရာကောင်းပါသည်](https://www.troyhunt.com/talking-with-scott-hanselman-on/)။ အသုံးပြုပြီးပါက ယင်းတို့ကို ပြန်လည်

ဖျက်ပစ်ရန် သေချာပါစေ၊ သို့မဟုတ်ပါက သင့်ဖုန်းသည် အမည်တူသော ကွန်ရက်တစ်ခုခုကို နောက်ပိုင်းတွင် ဝမ်းမြောက်ဝမ်းသာ ကြေညာ၍ ပြန်လည်ချိတ်ဆက်ပါလိမ့်မည်!

အကယ်၍ သင်သည် မယုံကြည်ရသော ကွန်ရက်တစ်ခုပေါ်တွင် ရောက်ရှိနေပါက VPN ကို အသုံးပြုခြင်းက အကျိုးရှိ *နိုင်ပါသည်*၊ သို့သော် သင်သည် VPN ဝန်ဆောင်မှုပေးသူကို *အလွန်* ယုံကြည်ရမည်ဖြစ်ကြောင်း သတိရပါ။ သင့် ISP ထက် ယင်းတို့ကို သင် တကယ် ပိုယုံကြည်ပါသလား။ အကယ်၍ သင်သည် VPN တစ်ခု အမှန်တကယ် လိုအပ်ပါက သင်ယုံကြည်ကြောင်း သေချာသည့် ဝန်ဆောင်မှုကို အသုံးပြုပါ၊ ထို့ပြင် ယင်း အတွက် ပိုက်ဆံပေး၍ သုံးသင့်သည်။ သို့မဟုတ်ပါက မိမိကိုယ်တိုင် **WireGuard** (https://www.wireguard.com/) တပ်ဆင်ပါ – ယင်းသည် အလွန်ကောင်းမွန်ပါသည် (https://web.archive.org/web/20210526211307/https://latacora.micro.blog/there-will-be/)!

cipherlist.eu (https://cipherlist.eu/) တွင် အင်တာနက် ချိတ်ဆက်ထားသော application များစွာအတွက် လုံခြုံသော ဆက်တင်ဖွဲ့စည်းမှုများလည်း ရှိပါသည်။ အကယ်၍ သင်သည် သီးသန့်လုံခြုံမှုကို အထူးဦးစားပေးပါက **privacytools.io** (https://privacytools.io) သည်လည်း အရင်းအမြစ်ကောင်းတစ်ခု ဖြစ်သည်။

သင်တို့အနက် အချို့သည် **Tor** (https://www.torproject.org/) အကြောင်း သိလိုကြပေမည်။ Tor သည် အင်အားကြီးမားသော ကမ္ဘာလုံးဆိုင်ရာ တိုက်ခိုက်သူများကို ခုခံနိုင်စွမ်း အထူး *မရှိပါ*၊ ထို့ပြင် Traffic analysis တိုက်ခိုက်မှုများကိုလည်း အားနည်းကြောင်း သတိရပါ။ ယင်းသည် သေးငယ်သော စကေးတွင် Traffic ကို ဖုံးကွယ်ရန် အသုံးဝင်နိုင်သော်လည်း သီးသန့်လုံခြုံမှုအတွက် များစွာ ကူညီပေးနိုင်မည် မဟုတ်ပါ။ ပိုမိုလုံခြုံသော ဝန်ဆောင်မှုများကို စတင်ကတည်းက အသုံးပြုခြင်းက ပိုမိုကောင်းမွန်ပါသည် (Signal, TLS + certificate pinning, စသည်ဖြင့်)။

ဝဘ် လုံခြုံရေး (Web Security)

ဒါဆို သင်က ဝဘ်ကိုပါ အသုံးပြုချင်သေးတာပေါ့? အို၊ သင် တကယ်ကို ကံစမ်းနေတာပဲ။

HTTPS Everywhere (https://www.eff.org/https-everywhere) ကို တပ်ဆင်ပါ။ SSL/TLS သည် အလွန်အရေးကြီးပါသည် (https://www.troyhunt.com/ssl-is-not-about-encryption/)၊ ယင်းသည် Encryption တစ်ခုတည်းတင် *မကဘဲ*၊ သင့်တော်သော ဝန်ဆောင်မှုနှင့် စကားပြောနေခြင်း ဟုတ်မဟုတ် ပထမဆုံး စိစစ်နိုင်ခြင်းလည်း ဖြစ်ပါသည်! အကယ်၍ သင်သည် ကိုယ်ပိုင် Web server ကို run နေပါက ယင်းကို စမ်းသပ်ပါ (https://www.ssllabs.com/ssltest/index.html)။ TLS ဆက်တင်ဖွဲ့စည်းမှုသည် ရှုပ်ထွေးနိုင်ပါသည် (https://wiki.mozilla.org/Security/Server_Side_TLS)။ HTTPS Everywhere သည် အခြားရွေးချယ်စရာ ရှိပါက သင့်ကို HTTP ဆိုက်များသို့ ဘယ်တော့မှ မရောက်ရှိစေရန် အစွမ်းကုန် ကြိုးစားပေးပါလိမ့်မည်။ ယင်းက သင့်ကို လုံးဝ ကယ်တင်နိုင်မည် မဟုတ်သော်လည်း ကူညီပေးနိုင်ပါသည်။ အကယ်၍ သင်သည် အမှန်တကယ် စိုးရိမ်တက်ပါက သင် အမှန်တကယ် မလိုအပ်သော SSL/TLS CA များကို Blacklist ထည့်ထားပါ။

uBlock Origin (https://github.com/gorhill/uBlock) ကို တပ်ဆင်ပါ။ ယင်းသည် ကြော်ငြာများကို သာမက စာမျက်နှာတစ်ခုမှ ပြုလုပ်ရန် ကြိုးစားနိုင်သော အခြား ပြင်ပဆက်သွယ်မှု အမျိုးအစား အားလုံးကို တားဆီးပေးသည့် **နယ်ပယ်ကျယ်ပြန့်သော တားဆီးမှုစနစ် (wide-spectrum blocker)** (https://github.com/gorhill/uBlock/wiki/Blocking-mode) ဖြစ်သည်။ Inline script များ စသည်တို့ကိုလည်း တားဆီးပေးသည်။ အကယ်၍ သင်သည် အရာရာ အဆင်ပြေစေရန် ဆက်တင်များ ပြင်ဆင်ရာတွင် အချိန်အနည်းငယ်ပေးနိုင်ပါက **medium mode** (https://github.com/gorhill/uBlock/wiki/Blocking-mode:-medium-mode) သို့မဟုတ် **hard mode** (https://github.com/gorhill/uBlock/wiki/Blocking-mode:-hard-mode) သို့ သွားပါ။ ထိုဆက်တင်များသည် ဆက်တင်များကို လုံလောက်စွာ မချိန်ညှိမချင်း အချို့ဝဘ်ဆိုက်များကို အလုပ်မလုပ်အောင် ပြုလုပ်သွားမည် *ဖြစ်သော်လည်း* သင့်အွန်လိုင်း လုံခြုံရေးကိုလည်း သိသိသာသာ တိုးတက်စေပါလိမ့်မည်။

အကယ်၍ သင်သည် Firefox ကို အသုံးပြုနေပါက **Multi-Account Containers** (https://support.mozilla.org/en-US/kb/containers) ကို ဖွင့်ပါ။ လူမှုကွန်ရက်များ၊ ဘဏ်လုပ်ငန်း၊ ဈေးဝယ်ခြင်း စသည်တို့အတွက် သီးခြား Container များ ဖန်တီးပါ။ Firefox သည် Container တစ်ခုစီအတွက် Cookies နှင့် အခြား State များကို လုံးဝသီးခြားစီ သိမ်းဆည်းထားမည်ဖြစ်သဖြင့် Container တစ်ခုတွင် သင်ဝင်ရောက်ကြည့်ရှုသော ဆိုက်များသည် အခြား Container များမှ အထိလွယ်သော ဒေတာများကို ချောင်းကြည့်နိုင်မည် မဟုတ်ပါ။ Google Chrome တွင် တူညီသော ရလဒ်များ ရရှိရန် **Chrome Profiles** (https://support.google.com/chrome/answer/2364824) ကို အသုံးပြုနိုင်သည်။

လေ့ကျင့်ခန်းများ (Exercises)

1. PGP အသုံးပြု၍ ဖိုင်တစ်ခုကို Encrypt ပြုလုပ်ပါ။
2. ရိုးရှင်းသော Encrypted volume တစ်ခု ဖန်တီးရန် veracrypt ကို အသုံးပြုပါ။
3. သင့်ဒေတာ အထိလွယ်ဆုံး အကောင့်များ (ဥပမာ GMail, Dropbox, Github, စသည်) အတွက် 2FA ကို ဖွင့်ပါ။

🔗 External Links & References (ပြင်ပ လင့်ခ်များနှင့် ကိုးကားချက်များ)

Resource / Description	URL	Micro QR
1. @TroyHunt	https://twitter.com/TroyHunt	
2. @SwiftOnSecurity	https://twitter.com/SwiftOnSecurity	
3. @taviso	https://twitter.com/taviso	
4. @thegrugq	https://twitter.com/thegrugq	
5. @tqbf	https://twitter.com/tqbf	
6. @mattblaze	https://twitter.com/mattblaze	
7. @moxie	https://twitter.com/moxie	
8. ဤစာရင်း (this list)	https://heimdalsecurity.com/blog/best-twitter-cybersec-accounts/	
9. သတင်းထောက်များအတွက် ပြုလုပ်ရန်နှင့် ရှောင်ကြဉ်ရန် အချက်များ (do's and don'ts for journalists)	https://web.archive.org/web/20221123204419/https://techsolidarity.org/resources/basic_security.htm	
10. @thegrugq	https://medium.com/@thegrugq	
11. ခရီးသွားလာရေး လုံခြုံရေး အကြံပြုချက် (travel security advice)	https://medium.com/@thegrugq/stop-fabricating-travel-security-advice-35259bf0e869	
12. USB သည် ကြောက်စရာ ကောင်းသောကြောင့်	https://www.bleepingcomputer.com/news/security/heres-a-list-of-29-different-types-of-usb-attacks/	
13. USB data blocker	https://www.amazon.com/dp/B00QRRZ2QM/	
14. 1password	https://1password.com/	
15. KeePass	https://keepass.info/	

Resource / Description	URL	Micro QR
16. BitWarden	https://bitwarden.com/	
17. `pass`	https://git.zx2c4.com/password-store/about/	
18. FIDO/U2F	https://fidoalliance.org/	
19. YubiKey	https://www.yubico.com/quiz/	
20. ကျောင်းသားများအတွက် ၂၀% လျှော့ဈေး	https://www.yubico.com/why-yubico/for-education/	
21. Phishing ကို ကာကွယ်မပေးနိုင်ပါ	https://twitter.com/taviso/status/1082015009348104192	
22. Signal	https://www.signal.org/	
23. တပ်ဆင်နည်း ညွှန်ကြားချက်များ	https://medium.com/@mshelton/signal-for-beginners-c6b44f76a1f0	
24. Wire	https://wire.com/en/	
25. သုံးနိုင်ပါသည်	https://www.securemessagingapps.com/	
26. Telegram ကို မသုံးပါနှင့်	https://twitter.com/bascule/status/897187286554628096	
27. keybase.io	https://keybase.io/	
28. ![$5 wrench	https://imgs.xkcd.com/comics/security.png	
29. cryptsetup + LUKS	https://wiki.archlinux.org/index.php/Dm-crypt/Encrypting_a_non-root_file_system	
30. BitLocker	https://fossbytes.com/enable-full-disk-encryption-windows-10/	
31. FileVault	https://support.apple.com/en-us/HT204837	

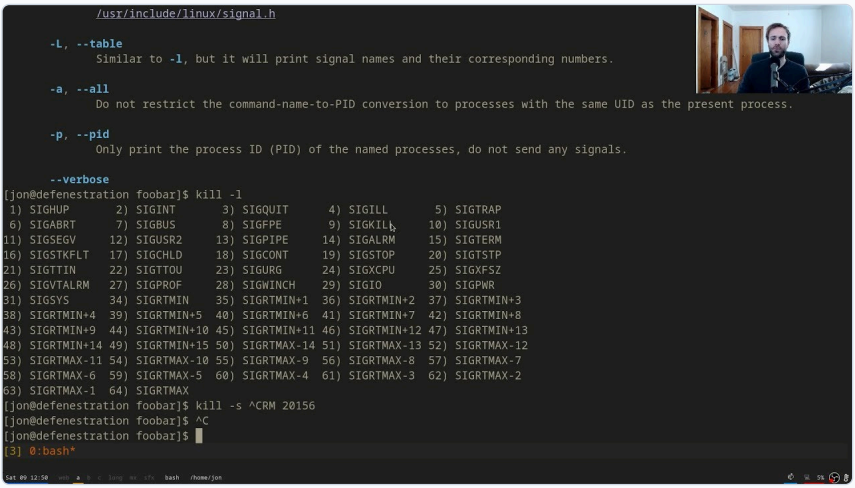
Resource / Description	URL	Micro QR
32. gocryptfs	https://github.com/rfjakob/gocryptfs	
33. eCryptFS	https://www.ecryptfs.org/	
34. ဤနေရာ	https://nuetzlich.net/gocryptfs/comparison/	
35. ဤနေရာ	https://wiki.archlinux.org/index.php/disk_encryption#Comparison_table	
36. ယုတ္တိတန့်စွာ ငြင်းဆိုနိုင်မှု (Plausible deniability)	https://en.wikipedia.org/wiki/Plausible_deniability	
37. deniable encryption	https://en.wikipedia.org/wiki/Deniable_encryption	
38. ဤနေရာရှိ ဆွေးနွေးချက်	https://security.stackexchange.com/questions/135846/is-plausible-deniability-actually-feasible-for-encrypted-volumes-disks	
39. VeraCrypt	https://www.veracrypt.fr/en/Home.html	
40. Tarsnap	https://www.tarsnap.com/	
41. Borgbase	https://www.borgbase.com/	
42. ကြောက်စရာကောင်းပါသည်	https://www.troyhunt.com/the-beginners-guide-to-breaking-website/	
43. ကြောက်စရာကောင်းပါသည်	https://www.troyhunt.com/talking-with-scott-hanselman-on/	
44. WireGuard	https://www.wireguard.com/	
45. အလွန်ကောင်းမွန်ပါသည်	https://web.archive.org/web/20210526211307/https://latacora.micro.blog/there-will-be/	
46. cipherlist.eu	https://cipherlist.eu/	
47. privacytools.io	https://privacytools.io	

Resource / Description	URL	Micro QR
48. Tor	https://www.torproject.org/	
49. HTTPS Everywhere	https://www.eff.org/https-everywhere	
50. အလွန်အရေးကြီးပါသည်	https://www.troyhunt.com/ssl-is-not-about-encryption/	
51. ယင်းကို စမ်းသပ်ပါ	https://www.ssllabs.com/ssltest/index.html	
52. ရှုပ်ထွေးနိုင်ပါသည်	https://wiki.mozilla.org/Security/Server_Side_TLS	
53. uBlock Origin	https://github.com/gorhill/uBlock	
54. နယ်ပယ်ကျယ်ပြန့်သော တားဆီးမှုစနစ် (wide-spectrum blocker)	https://github.com/gorhill/uBlock/wiki/Blocking-mode	
55. medium mode	https://github.com/gorhill/uBlock/wiki/Blocking-mode:-medium-mode	
56. hard mode	https://github.com/gorhill/uBlock/wiki/Blocking-mode:-hard-mode	
57. Multi-Account Containers	https://support.mozilla.org/en-US/kb/containers	
58. Chrome Profiles	https://support.google.com/chrome/answer/2364824	

Shell နှင့် Script ရေးသားခြင်း

▶ LECTURE VIDEO REFERENCE

Shell နှင့် Script ရေးသားခြင်း



Scan the QR code below using your mobile camera or click the link to watch this full lecture online:

Direct Online Link:

https://www.youtube.com/watch?v=dbDRfmH5uSI

Shell ဆိုသည်မှာ သင်၏ ကွန်ပျူတာကို စာသားဖြင့် ထိရောက်စွာ ခိုင်းစေနိုင်သည့် Interface တစ်ခု ဖြစ်ပါသည်။

Shell prompt ဆိုသည်မှာ Terminal တစ်ခုကို ဖွင့်လိုက်သည့်အခါ သင့်အား ကြိုဆိုသည့် နေရာဖြစ်ပါသည်။ ယင်းက သင့်အား ပရိုဂရမ်များနှင့် Command များကို Run နိုင်စေပါသည်။ အသုံးများသော Command အချို့မှာ-

- Directory ပြောင်းလဲရန် `cd`
- ဖိုင်များနှင့် Directory များကို စာရင်းထုတ်ကြည့်ရန် `ls`
- ဖိုင်များကို ရွှေ့ပြောင်းရန်နှင့် ကူးယူရန် `mv` နှင့် `cp`

သို့သော် Shell သည် ထိုမျှမက *အဆမတန်* ပိုမို ပြုလုပ်နိုင်ပါသည်။ သင့် ကွန်ပျူတာပေါ်ရှိ မည်သည့် ပရိုဂရမ်ကိုမဆို ခေါ်ယူ အသုံးပြုနိုင်ပြီး၊ သင် ပြုလုပ်လိုသမျှ ကိစ္စတိုင်းနီးပါးအတွက် Command-line tool များ ရှိကြပါသည်။ ထို့အပြင် ၎င်းတို့သည် Graphical (GUI) နည်းလမ်းများထက် ပိုမို အလုပ်တွင် မြန်ဆန်လေ့ ရှိကြပါသည်။ ဤသင်ခန်းစာတွင် ယင်း tool အများအပြားကို လေ့လာသွားမည် ဖြစ်ပါသည်။

Shell သည် အပြန်အလှန် တုံ့ပြန်နိုင်သော Programming Language (“scripting”) တစ်ခုကို ပံ့ပိုးပေးပါသည်။ Shell အမျိုးအစားများစွာ ရှိပါသည်-

- သင်သည် `sh` သို့မဟုတ် `bash` ကို အသုံးပြုဖူးပေလိမ့်မည်။
- ပရိုဂရမ်းမင်း ဘာသာစကားများနှင့် စတိုင်တူသော Shell များလည်း ရှိသည်- ဥပမာ `csh` ။
- သို့မဟုတ် ပိုမို “ကောင်းမွန်သော” Shell များ- `fish` , `zsh` , `ksh` ။

ဤသင်ခန်းစာတွင် နေရာတိုင်း၌ တွေ့ရှိနိုင်သော `sh` နှင့် `bash` ကို အဓိကထားသွားမည် ဖြစ်သော်လည်း အခြား Shell များကိုလည်း စမ်းသပ်ကြည့်နိုင်ပါသည်။ ကျွန်တော်ကတော့ `fish` ကို ကြိုက်ပါသည်။

Shell programming သည် သင်၏ tool အိတ်ထဲတွင် *အလွန်* အသုံးဝင်သော tool တစ်ခု ဖြစ်ပါသည်။ ပရိုဂရမ်များကို Prompt တွင် တိုက်ရိုက် ရေးသားနိုင်သကဲ့သို့ ဖိုင်တစ်ခုအတွင်းသို့လည်း ရေးသားနိုင်ပါသည်။ Shell ကို Execute လုပ်နိုင်ရန်အတွက် `#!/bin/sh` + `chmod +x` ကို အသုံးပြုပါသည်။

Shell ကို အသုံးပြု၍ အလုပ်လုပ်ခြင်း

Command တစ်ခုကို အကြိမ်များစွာ Run ခြင်း-

```
for i in $(seq 1 5); do echo hello; done
```

ဒီနေရာမှာ အသေးစိတ် လေ့လာစရာများစွာ ရှိပါသည်-

- `for x in list; do BODY; done`
  - `;` သည် Command တစ်ခုကို အဆုံးသတ်ပေးသည် – စာကြောင်းအသစ် (newline) ဖြတ်လိုက်သကဲ့သို့ တူညီသည်
  - `list` ကို ခွဲခြားပြီး၊ တစ်ခုစီကို `x` တွင် သတ်မှတ်ကာ body ကို Run သည်

- ခွဲခြားခြင်းသည် “whitespace splitting” (ဟာကွက်ဖြင့် ခွဲခြားခြင်း) ဖြစ်ပြီး နောက်တွင် ထပ်မံ ရှင်းပြပါမည်
- Shell တွင် Curly braces ({}) များ မပါရှိသဖြင့် `do` + `done` ကို အသုံးပြုသည်

- `$(seq 1 5)`
  - `seq` ပရိုဂရမ်ကို Argument များဖြစ်သော `1` နှင့် `5` ဖြင့် Run သည့် သဘောဖြစ်သည်
  - `$()` တစ်ခုလုံးကို ထိုပရိုဂရမ်၏ Output ဖြင့် အစားထိုးသည်
  - အောက်ပါအတိုင်း ရေးသားခြင်းနှင့် တူညီသည်-

    ```
    for i in 1 2 3 4 5
    ```

- `echo hello`
  - Shell script ထဲရှိ အရာအားလုံးသည် Command များ ဖြစ်ကြသည်
  - ဤနေရာတွင် မိမိထံ ပေးပို့လိုက်သော Argument များကို ရိုက်နှိပ်ပြသသည့် `echo` command ကို `hello` ဆိုသည့် Argument ဖြင့် Run ခြင်း ဖြစ်သည်
  - Command အားလုံးကို Colon (:) ဖြင့် ခွဲခြားထားသော `$PATH` အတွင်း၌ ရှာဖွေသည်

Variable များလည်း ရှိကြပါသည်-

```
for f in $(ls); do echo $f; done
```

ယင်းက လက်ရှိ Directory ထဲရှိ ဖိုင်အမည် တစ်ခုစီကို ရိုက်နှိပ်ပြသမည် ဖြစ်ပါသည်။ Variable များကို `=` (Space မပါဘဲ!) အသုံးပြု၍လည်း သတ်မှတ်နိုင်ပါသည်-

```
foo=bar
echo $foo
```

“သီးသန့်” Variable အများအပြားလည်း ရှိပါသည်-

- `$1` မှ `$9` : Script ထံ ပေးပို့သော Argument များ
- `$0` : Script ၏ အမည်ကိုယ်တိုင်
- `$#` : Argument အရေအတွက်

- `$$` : လက်ရှိ Shell ၏ Process ID (PID)

Directory များကိုသာ ရိုက်နှိပ်ပြသလိုပါက-

```
for f in $(ls); do if test -d $f; then echo dir $f; fi; done
```

ဒီနေရာမှာလည်း အသေးစိတ် လေ့လာစရာများ ပါရှိပါသည်-

- `if CONDITION; then BODY; fi`
  - `CONDITION` သည် Command တစ်ခု ဖြစ်ပြီး ယင်းက Exit status 0 (အောင်မြင်မှု) ဖြင့် ပြန်လာပါက `BODY` ကို Run ပေးသည်
  - `else` သို့မဟုတ် `elif` များကိုလည်း ချိတ်ဆက် အသုံးပြုနိုင်သည်
  - ဤနေရာတွင်လည်း Curly braces မပါရှိသဖြင့် `then` + `fi` ကို အသုံးပြုသည်
- `test` သည် ဆန်းစစ်မှုများနှင့် နှိုင်းယှဉ်မှု အမျိုးမျိုးကို ပြုလုပ်ပေးသည့် အခြား ပရိုဂရမ်တစ်ခု ဖြစ်ပြီး မှန်ကန်ပါက 0 ဖြင့် Exit လုပ်သည် (`$?`)
  - `man COMMAND` သည် သင့်အတွက် အကူအညီဖြစ်သည်- `man test`
  - `[` + `]` ဖြင့်လည်း ခေါ်ယူ အသုံးပြုနိုင်သည်- `[ -d $f ]` ```bash
- `man test` နှင့် `which "["` ကို ကြည့်ရှုပါ

```
ဒါပေမဲ့ ခဏနေပါဦး! ဒါ မှားနေပါတယ်! ဖိုင်အမည်က "My Documents" လို့ ခေါ်ရင် ဘာဖြစ်မလဲ?

 - `for f in $(ls)` သည် `for f in My Documents` အဖြစ် ဖြန့်ကားသွားမည် ဖြစ်သည်
 - ပထမဦးစွာ `My` ကို စစ်ဆေးပြီး၊ ထို့နောက် `Documents` ကို စစ်ဆေးပါလိမ့်မည်
 - ဒါဟာ ကျွန်ုပ်တို့ လိုချင်တဲ့ အရာ မဟုတ်ပါဘူး!
 - ဒါဟာ Shell script တွေမှာ Bug တွေ ဖြစ်ပေါ်စေတဲ့ အကြီးမားဆုံး အကြောင်းအရင်း ဖြစ်ပါတယ်

## Argument များကို ခွဲခြားခြင်း (Argument splitting)

Bash သည် Argument များကို Whitespace (ဟာကွက်) ဖြင့် ခွဲခြားပါသည်။ ဒါဟာ သင် အမြဲတမ်း လိုလားတဲ့ အရာ မဟုတ်နိုင်ပါဘူး!

 - Argument ထဲရှိ Space များကို ကိုင်တွယ်ရန် Quoting (မျက်တောင်အဖွင့်အပိတ်) အသုံးပြုဖို့ လိုအပ်သည်၊ `for f in "My Documents"` ဆိုလျှင် မှန်ကန်စွာ အလုပ်လုပ်မည် ဖြစ်သည်
 - အခြားနေရာတစ်ခုမှာလည်း ဒီပြဿနာမျိုး ရှိနေသည် -- ဘယ်နေရာလဲဆိုတာ မြင်ပါသလား?
   `test -d $f`: အကယ်၍ `$f` တွင် Whitespace ပါဝင်နေပါက `test` သည် Error တက်ပါလိမ့်မည်!
 - `echo` ကတော့ Space ဖြင့် ခွဲခြားပြီး ပြန်ပေါင်းပေးတာကြောင့် အဆင်ပြေနေသလို ရှိသော်လည်း ဖိုင်အမည်ထဲတွင် Newline ပါဝင်နေရင် ဘာဖြစ်မလဲ?! Space အဖြစ် ပြောင်းလဲသွားပါလိမ့်မည်!
 - မခွဲခြားစေလိုသော Variable အသုံးပြုမှု တိုင်းကို Quoting ဖြင့် အုပ်ပေးပါ
 - ဒါပေမဲ့ အထက်ပါ ကျွန်ုပ်တို့၏ Script ကို ဘယ်လို ပြင်ကြမလဲ?
   `for f in "$(ls)"` က ဘာလုပ်ပေးမယ်လို့ ထင်ပါသလဲ?

Globbing (Pattern ဖြင့် ရှာဖွေခြင်း) သည် ယင်း၏ အဖြေဖြစ်ပါသည်။

 - Bash သည် Pattern များကို အသုံးပြု၍ ဖိုင်များကို မည်သို့ ရှာဖွေရမည်ကို သိရှိသည်-
   - `*` မည်သည့် စာလုံးစုမဆို
   - `?` မည်သည့် စာလုံးတစ်လုံးမဆို
   - `{a,b,c}` ဤစာလုံးများထဲမှ မည်သည့် စာလုံးမဆို
 - `for f in *`: ဤ Directory ထဲရှိ ဖိုင်အားလုံး
 - Globbing ပြုလုပ်သည့်အခါ ကိုက်ညီသော ဖိုင်တစ်ခုစီသည် သီးခြား Argument တစ်ခု ဖြစ်လာသည်
   - သို့သော် _အသုံးပြုသည့်အခါ_ Quoting အုပ်ရန် လိုအပ်ဆဲဖြစ်သည်- `test -d "$f"`
 - ပိုမို အဆင့်မြင့်သော Pattern များကို ဖန်တီးနိုင်သည်-
   - `for f in a*`: လက်ရှိ Directory ထဲရှိ `a` ဖြင့် စသော ဖိုင်အားလုံး
   - `for f in foo/*.txt`: `foo` ထဲရှိ `.txt` ဖိုင်အားလုံး
   - `for f in foo/*/p??.txt`: `foo` ၏ Subdirectory များထဲရှိ `p` ဖြင့် စသော စာလုံးသုံးလုံးပါ စာသားဖိုင်များအားလုံး

Whitespace ပြဿနာများက ထိုမျှနှင့် မပြီးဆုံးသေးပါ-

 - `if [ $foo = "bar" ]; then` -- ပြဿနာကို မြင်ပါသလား?
 - အကယ်၍ `$foo` က ဗလာဖြစ်နေရင် ဘာဖြစ်မလဲ? `[` ထံ ပေးပို့သော Argument များသည် `=` နှင့် `bar` သာ ဖြစ်သွားမည်...
 - ဤသည်ကို `[ x$foo = "xbar" ]` ဖြင့် ဖြေရှင်းနိုင်သော်လည်း သိပ်မကောင်းလှပါ
 - ယင်းအစား သီးသန့် Parsing ပါဝင်သော Bash built-in comparator ဖြစ်သည့် `[[` ကို အသုံး
```

```
ပြုပါ
   - ထို့အပြင် `-a` အစား `&&` ကိုလည်းကောင်း၊ `-o` အစား `||` ကိုလည်းကောင်း စသည်ဖြင့် အ
သုံးပြုခွင့် ပေးသည်

<!-- TODO: arrays? $@. ${array[@]} vs "${array[@]}". -->

## ပေါင်းစပ် အသုံးပြုနိုင်မှု (Composability)

Shell သည် စွမ်းဆောင်ရည် ထက်မြက်ရခြင်း၏ အကြောင်းရင်း အစိတ်အပိုင်းတစ်ခုမှာ ပေါင်းစပ် အသုံးပြုနို
င်မှုကြောင့် ဖြစ်ပါသည်။ အရာအားလုံးကို ပြုလုပ်သည့် ပရိုဂရမ်တစ်ခုတည်း ရှိမည့်အစား ပရိုဂရမ် အများအပြာ
းကို အပြန်အလှန် ချိတ်ဆက် အသုံးပြုနိုင်ပါသည်။

အဓိက သင်္ကေတမှာ `|` (pipe) ဖြစ်ပါသည်။

 - `a | b` ဆိုသည်မှာ `a` နှင့် `b` နှစ်ခုလုံးကို Run ပြီး၊ `a` ၏ Output အားလုံးကို `b` ၏ Inp
ut အဖြစ် ပို့ဆောင်ကာ `b` ၏ Output ကို ရိုက်နှိပ်ပြသခြင်း ဖြစ်သည်

သင် စတင်လိုက်သော ပရိုဂရမ်တိုင်း ("processes") တွင် "stream" သုံးခု ပါရှိကြသည်-

 - `STDIN`: ပရိုဂရမ်က Input ကို ဖတ်ရှုသည့်အခါ ဤနေရာမှ လာသည်
 - `STDOUT`: ပရိုဂရမ်က တစ်ခုခု ရိုက်နှိပ်ပြသသည့်အခါ ဤနေရာသို့ သွားသည်
 - `STDERR`: ပရိုဂရမ်က အသုံးပြုရန် ရွေးချယ်နိုင်သော ဒုတိယမြောက် Output ဖြစ်သည်
 - ပုံမှန်အားဖြင့် `STDIN` သည် သင်၏ Keyboard ဖြစ်ပြီး `STDOUT` နှင့် `STDERR` နှစ်ခုလုံးသည် သ
င်၏ Terminal ဖြစ်ကြသည်၊ သို့သော် ယင်းကို ပြောင်းလဲနိုင်ပါသည်!
   - `a | b` သည် `a` ၏ `STDOUT` ကို `b` ၏ `STDIN` အဖြစ် ပြောင်းလဲပေးသည်
   - အောက်ပါအတိုင်းလည်း ရှိကြသည်-
```bash
 - `a > foo` (`a` ၏ `STDOUT` သည် `foo` ဖိုင်သို့ သွားသည်)
 - `a 2> foo` (`a` ၏ `STDERR` သည် `foo` ဖိုင်သို့ သွားသည်)
 - `a < foo` (`a` ၏ `STDIN` ကို `foo` ဖိုင်မှ ဖတ်ရှုသည်)
 - အကြံပြုချက်- `tail -f` သည် ဖိုင်တစ်ခုအတွင်းသို့ ရေးသားနေစဉ် ယင်းဖိုင်ကို ရိုက်နှိပ်ပြသမည် ဖြစ်
သည်
```

- ဒါဟာ ဘာကြောင့် အသုံးဝင်သနည်း? ပရိုဂရမ်တစ်ခု၏ Output ကို စိတ်ကြိုက် ပြုပြင်ပြောင်းလဲနိုင်စေသော ကြောင့်ဖြစ်သည်!
  - `ls | grep foo` : `foo` စကားလုံး ပါဝင်သော ဖိုင်များအားလုံး
  - `ps | grep foo` : `foo` စကားလုံး ပါဝင်သော Process များအားလုံး
  - `journalctl | grep -i intel | tail -n5` : intel စကားလုံးပါသော အဖျားပိုင်း စနစ် Log မက် ဆေ့ချ် ၅ ခု (စာလုံး အသေးအကြီး မခွဲခြားပါ)
  - `who | sendmail -t me@example.com` : Log in ဝင်ထားသော အသုံးပြုသူ စာရင်းကို `me@example.com` သို့ ပေးပို့သည်
```

- နောက်ပိုင်းတွင် ဖော်ပြမည့် Data-wrangling အများအပြားအတွက် အခြေခံ ဖြစ်လာသည်

Bash သည် ပရိုဂရမ်များကို ပေါင်းစပ်ရန် အခြား နည်းလမ်းများစွာကိုလည်း ပံ့ပိုးပေးထားပါသည်။

Command များကို `(a; b) | tac` ဖြင့် အစုဖွဲ့နိုင်ပါသည်- `a` ကို Run ပြီး ထို့နောက် `b` ကို Run ကာ ၎င်းတို့၏ Output အားလုံးကို Input များကို ပြောင်းပြန် ရိုက်နှိပ်ပေးသည့် `tac` ထံ ပို့ဆောင်သည်။

လူသိနည်းသော်လည်း အလွန် အသုံးဝင်သော နည်းလမ်းတစ်ခုမှာ *process substitution* ဖြစ်ပါသည်။ `b <(a)` သည် `a` ကို Run ပြီး ၎င်း၏ Output stream အတွက် ယာယီ ဖိုင်အမည်တစ်ခု ဖန်တီးကာ ထိုဖိုင်အမည်ကို `b` ထံ ပေးပို့မည် ဖြစ်သည်။ ဥပမာ-

```
diff <(journalctl -b -1 | head -n20) <(journalctl -b -2 | head -n20)
```

ယင်းက အရင်ကွန်ပျူတာ ပွင့်ခဲ့စဉ် (boot log) ၏ ပထမ စာကြောင်း ၂၀ နှင့် ယင်းမတိုင်မီက boot log တို့ အကြား ကွဲပြားချက်များကို ပြသပေးမည် ဖြစ်ပါသည်။

Job နှင့် Process များကို ထိန်းချုပ်ခြင်း

အချိန်ကြာမြင့်စွာ အလုပ်လုပ်ရမည့် အရာများကို နောက်ကွယ် (background) တွင် Run လိုပါက ဘာလုပ်ရမည်နည်း?

- `&` နောက်ဆက်တွဲ သင်္ကေတသည် ပရိုဂရမ်တစ်ခုကို “background” တွင် Run ပေးသည်
  - ယင်းက သင့်အား Prompt ကို ချက်ချင်း ပြန်လည် ပေးအပ်ပါလိမ့်မည်
  - Server နှင့် Client ကဲ့သို့ ပရိုဂရမ် နှစ်ခုကို တစ်ပြိုင်နက်တည်း Run လိုပါက အသုံးဝင်သည်- `server & client`
  - Run နေသော ပရိုဂရမ်သည် သင်၏ Terminal ကို `STDOUT` အဖြစ် အသုံးပြုနေဆဲ ဖြစ်သည်ကို သတိပြုပါ! `server > server.log & client` ကို စမ်းသပ်ကြည့်ပါ
- ထိုသို့သော Process အားလုံးကို `jobs` ဖြင့် ကြည့်ရှုနိုင်သည်
  - ယင်းက “Running” ဟု ပြသနေသည်ကို သတိပြုပါ
- ယင်းကို Foreground သို့ ပြန်ယူဆောင်ရန် `fg %JOB` ကို အသုံးပြုပါ (Argument မပါလျှင် နောက်ဆုံး တစ်ခုဖြစ်သည်)
- လက်ရှိ ပရိုဂရမ်ကို Background သို့ ပို့လိုပါက- `^Z` + `bg` (ဤနေရာတွင် `^Z` ဆိုသည်မှာ `Ctrl+Z` နှိပ်ခြင်း ဖြစ်သည်)

  - `^Z` သည် လက်ရှိ Process ကို ရပ်တန့်ပြီး “job” တစ်ခု အဖြစ် ပြုလုပ်ပေးသည်
  - `bg` သည် နောက်ဆုံး job ကို Background တွင် Run ပေးသည် (`&` ထည့်ထားသကဲ့သို့)
- Background job များကို လက်ရှိ Session တွင် ချိတ်ဆက်ထားဆဲ ဖြစ်ပြီး Log out ထွက်ပါက ပိတ်သွားမည် ဖြစ်သည်၊ `disown` သည် ထိုချိတ်ဆက်မှုကို ဖြတ်တောက်ပေးသည် သို့မဟုတ် `nohup` ကို အသုံးပြုနိုင်သည်
- `$!` သည် နောက်ဆုံး Background process ၏ PID ဖြစ်သည်

သင့် ကွန်ပျူတာပေါ်တွင် Run နေသော အခြား အရာများအတွက် ဘာလုပ်နိုင်သနည်း?

- `ps` သည် သင့်အတွက် အကူအညီ ဖြစ်သည်- Run နေသော Process များကို စာရင်းထုတ်ပေးသည်
  - `ps -A` : အသုံးပြုသူ အားလုံး၏ Process များကို ပြသသည် (`ps ax` လည်း ရသည်)
  - `ps` တွင် Argument များ *စွာ* ရှိပါသည်- `man ps` ကို ကြည့်ပါ
- `pgrep` : ရှာဖွေခြင်းဖြင့် Process များကို ရှာဖွေပေးသည် (`ps -A | grep` ကဲ့သို့)
  - `pgrep -af` : Argument များနှင့်တကွ ရှာဖွေပြသသည်
- `kill` : Process အား ID ဖြင့် *signal* ပေးပို့သည် (ရှာဖွေ၍ `-f` ဖြင့် ပေးပို့ရန် `pkill`)
  - Signal များသည် Process ကို “တစ်ခုခု ပြုလုပ်ရန်” ခိုင်းစေခြင်း ဖြစ်သည်
  - အသုံးအများဆုံးမှာ: `SIGKILL` (`-9` သို့မဟုတ် `-KILL`): *ချက်ချင်း* Exit လုပ်ရန် ခိုင်းစေသည် (`^\` နှင့် တူညီသည်)
  - ထို့အပြင် `SIGTERM` (`-15` သို့မဟုတ် `-TERM`): ပုံမှန်အတိုင်း ပြေပြစ်စွာ Exit လုပ်ရန် ခိုင်းစေသည် (`^C` နှင့် တူညီသည်)

Flag များ

Command line utility အများစုသည် Parameter များကို **flag** များ အသုံးပြု၍ ရယူကြသည်။ Flag များသည် အများအားဖြင့် အတိုပုံစံ (`-h`) နှင့် အရှည်ပုံစံ (`--help`) ဟူ၍ လာလေ့ရှိသည်။ ပုံမှန်အားဖြင့် `CMD -h` သို့မဟုတ် `man CMD` ကို Run ခြင်းက ပရိုဂရမ်က လက်ခံသည့် Flag စာရင်းကို ပေးပါလိမ့်မည်။ Short flag များကို ပေါင်းစပ်နိုင်လေ့ ရှိသည်၊ `rm -r -f` ကို Run ခြင်းသည် `rm -rf` သို့မဟုတ် `rm -fr` ကို Run ခြင်းနှင့် တူညီပါသည်။ အသုံးများသော Flag အချို့သည် စံနှုန်းတစ်ခုကဲ့သို့ ဖြစ်နေပြီး Application အများအပြားတွင် ၎င်းတို့ကို တွေ့ရမည် ဖြစ်သည်-

- `-a` သည် ပုံမှန်အားဖြင့် ဖိုင်အားလုံးကို ညွှန်းဆိုသည် (ဆိုလိုသည်မှာ Dot (.) ဖြင့် စသော ဖိုင်များ ပါဝင်သည်)

- `-f` သည် ပုံမှန်အားဖြင့် တစ်ခုခုကို အတင်းအကျပ် ပြုလုပ်ခြင်း (Force) ကို ညွှန်းဆိုသည်၊ ဥပမာ `rm -f`
- `-h` သည် Command အများစုအတွက် အကူအညီ (Help) ကို ပြသပေးသည်
- `-v` သည် ပုံမှန်အားဖြင့် အသေးစိတ် Output (Verbose) ကို ပေးသည်
- `-V` သည် ပုံမှန်အားဖြင့် Command ၏ Version ကို ရိုက်နှိပ်ပြသသည်

ထို့အပြင် Double dash `--` ကို Built-in command များနှင့် အခြား Command အများအပြားတွင် Command Option များ ပြီးဆုံးကြောင်း ဖော်ပြရန် အသုံးပြုပြီး ယင်းနောက်တွင် Positional parameter များကိုသာ လက်ခံတော့မည် ဖြစ်သည်။ ထို့ကြောင့် သင့်တွင် `-v` အမည်ရှိ ဖိုင်တစ်ခု ရှိပါက (ဖန်တီးနိုင်ပါသည်) ထိုဖိုင်ကို grep လုပ်လိုလျှင် `grep pattern -- -v` က အလုပ်လုပ်မည် ဖြစ်ပြီး `grep pattern -v` က အလုပ်လုပ်မည် မဟုတ်ပါ။ အမှန်တော့ ထိုကဲ့သို့သော ဖိုင်တစ်ခုကို ဖန်တီးသည့် နည်းလမ်းတစ်ခုမှာ `touch -- -v` ပြုလုပ်ခြင်း ဖြစ်ပါသည်။

လေ့ကျင့်ခန်းများ

1. သင်သည် Shell ကို လုံးဝ အသစ်စတင် အသုံးပြုသူ ဖြစ်ပါက **BashGuide** (https://mywiki.wooledge.org/BashGuide) ကဲ့သို့သော ပိုမို ပြည့်စုံသည့် လမ်းညွှန်ကို ဖတ်ရှုလိုပေလိမ့်မည်။ အကယ်၍ ပိုမို နက်နဲသော မိတ်ဆက်ကို လိုချင်ပါက **The Linux Command Line** (https://linuxcommand.org/tlcl.php) သည် အရင်းအမြစ် ကောင်းတစ်ခု ဖြစ်ပါသည်။

2. **PATH, which, type**

```
Command line မှတစ်ဆင့် Run သည့် ပရိုဂရမ်များကို ရှာဖွေရန် `PATH` environment variable ကို
အသုံးပြုကြောင်း အနည်းငယ် ဆွေးနွေးခဲ့ကြပြီး ဖြစ်ပါသည်။ ၎င်းကို အနည်းငယ် ထပ်မံ လေ့လာကြည့်ကြစို့-
- `echo $PATH` (သို့မဟုတ် လှပစွာ ပြသရန် `echo $PATH | tr -s ':' '\n'`) ကို Run ပြီး ၎င
းး၏ ပါဝင်မှုများကို စစ်ဆေးပါ၊ မည်သည့် တည်နေရာများ စာရင်းပါဝင်သနည်း?
- `which` command သည် User PATH အတွင်း၌ ပရိုဂရမ်တစ်ခုကို ရှာဖွေပေးသည်။ `echo`, `ls`
သို့မဟုတ် `mv` ကဲ့သို့သော အသုံးများသည့် Command များအတွက် `which` ကို Run ကြည့်ပါ။ `which
` သည် Shell alias များကို နားမလည်သောကြောင့် အနည်းငယ် ကန့်သတ်ချက် ရှိသည်ကို သတိပြုပါ၊
ထို Command များအတွက် `type` နှင့် `command -v` များကို Run ကြည့်ပါ။ Output မည်သို့ ကွဲပြား
သနည်း?
- `PATH=` ကို Run ပြီး ယခင် Command များကို ထပ်မံ Run ကြည့်ပါ၊ အချို့ အလုပ်လုပ်ပြီး အချို့ အ
လုပ်မလုပ်ပါ၊ ဘာကြောင့်လဲဆိုတာ သင် ရှာဖွေနိုင်ပါသလား?
```

1. **သီးသန့် Variable များ**

```
- `~` variable သည် မည်သို့ ဖြန့်ကားသနည်း? `.` ကော ဘာလဲ? `..` ကော ဘာလဲ?
- `$$?` variable က ဘာလုပ်ပေးသနည်း?
- `$_` variable က ဘာလုပ်ပေးသနည်း?
- `!!` variable သည် မည်သည့်အရာအဖြစ် ဖြန့်ကားသနည်း? `!!*` ကော ဘာလဲ? `!l` ကော ဘာလဲ?
- ဤ Option များအတွက် Documentation ကို ရှာဖွေပြီး ရင်းနှီးကျွမ်းဝင်အောင် ပြုလုပ်ပါ
```

1. **xargs**

```
အချို့အချိန်များတွင် Pipe ပြုလုပ်ခံရသော Command သည် စာကြောင်းအသစ် (newline) ခွဲထားသည့် Format ကို မျှော်လင့်မထားသောကြောင့် Piping သည် အဆင်မပြေဖြစ်တတ်ပါသည်။ ဥပမာ `file` command သည် ဖိုင်၏ Property များကို ပြသပေးသည်။

`ls | file` နှင့် `ls | xargs file` တို့ကို Run ကြည့်ပါ။ `xargs` က ဘာလုပ်နေသနည်း?
```

1. **Shebang**

```
Script တစ်ခုကို ရေးသားသည့်အခါ <a href="https://en.wikipedia.org/wiki/Shebang_(Unix" class="ext-link">shebang</a> <span class="ext-url-print">(https://en.wikipedia.org/wiki/Shebang_(Unix)</span>) စာကြောင်းကို အသုံးပြု၍ Script ကို အဓိပ္ပာယ်ပြန်ဆိုရန် မည်သည့် Interpreter ကို အသုံးပြုရမည်ကို သင်၏ Shell ထံ သတ်မှတ်ပေးနိုင်ပါသည်။ အောက်ပါ ပါဝင်မှုများဖြင့် `hello` အမည်ရှိ Script တစ်ခုကို ရေးသားပြီး `chmod +x hello` ဖြင့် Execute လုပ်နိုင်အောင် ပြုလုပ်ပါ။ ထို့နောက် `./hello` ဖြင့် Execute လုပ်ပါ။ ထို့နောက် ပထမစာကြောင်းကို ဖျက်ပြီး နောက်တစ်ကြိမ် ထပ်မံ Execute လုပ်ပါ? Shell သည် ထိုပထမစာကြောင်းကို မည်သို့ အသုံးပြုနေသနည်း?
```

```
```bash
  #! /usr/bin/python

  print("Hello World!")
```
```

`#! usr/bin/env bash` ကဲ့သို့သော Shebang ပါဝင်သည့် ပရိုဂရမ်များကို မကြာခဏ တွေ့ရပါလိမ့်မည်။ ဤသည်မှာ ၎င်း၏ ကိုယ်ပိုင် အားသာချက်၊ အားနည်းချက်များ (https://unix.stackexchange.com/questions/29608/why-is-it-better-to-use-usr-bin-env-name-instead-of-path-to-name-as-my) ရှိသည့် ပိုမို Portable ဖြစ်သော ဖြေရှင်းချက် ဖြစ်ပါသည်။ `env` သည် `which` နှင့် မည်သို့ ကွဲပြားသနည်း? မည်သည့် ပရိုဂရမ်ကို Run ရမည်ကို ဆုံးဖြတ်ရန် `env` က မည်သည့် Environment variable ကို အသုံးပြုသနည်း?

1. **Pipes, process substitution, subshell**

အောက်ပါ ပါဝင်မှုများဖြင့် `slow_seq.sh` အမည်ရှိ Script တစ်ခုကို ဖန်တီးပြီး Execute လုပ်နိုင်ရန် `chmod +x slow_seq.sh` ပြုလုပ်ပါ။

```bash
  #! /usr/bin/env bash

  for i in $(seq 1 10); do
          echo $i;
          sleep 1;
  done
```

Pipe များ (နှင့် Process substitution) သည် Subshell execution သို့မဟုတ် `$()` အသုံးပြုခြင်းနှင့် ကွဲပြားသည့် နည်းလမ်းတစ်ခု ရှိသည်။ အောက်ပါ Command များကို Run ပြီး ကွဲပြားချက်များကို လေ့လာကြည့်ပါ-

- `./slow_seq.sh | grep -P "[3-6]"`
- `grep -P "[3-6]" <(./slow_seq.sh)`
- `echo $(./slow_seq.sh) | grep -P "[3-6]"`

1. **အထွေထွေ (Misc)**

- `touch {a,b}{a,b}` ပြီးလျှင် `ls` ကို Run ကြည့်ပါ၊ ဘာပေါ်လာသနည်း?
- အချို့အချိန်များတွင် STDIN ကို ထိန်းသိမ်းထားပြီး ဖိုင်တစ်ခုသို့ Pipe ပို့လိုကြသည်။ `echo HELLO | tee hello.txt` ကို Run ကြည့်ပါ
- `cat hello.txt > hello.txt ` ကို Run ကြည့်ပါ၊ ဘာဖြစ်လာမည်ဟု မျှော်လင့်သနည်း? အမှန်တကယ် ဘာဖြစ်သွားသနည်း?
- `echo HELLO > hello.txt` ကို Run ပြီး ထို့နောက် `echo WORLD >> hello.txt` ကို Run ပါ။ `hello.txt` ၏ ပါဝင်မှုများမှာ မည်သည်တို့ ဖြစ်သနည်း? `>` သည် `>>` နှင့် မည်သို့ ကွဲပြားသနည်း?
- `printf "\e[38;5;81mfoo\e[0m\n"` ကို Run ပါ။ Output မည်သို့ ကွဲပြားသွားသနည်း? ပိုမို သိရှိလိုပါက ANSI color escape sequences ကို ရှာဖွေပါ။
- `touch a.txt` ကို Run ပြီး ထို့နောက် `^txt^log` ကို Run ပါ၊ bash က သင့်အတွက် ဘာလုပ်ပေးခဲ့သနည်း? ထို့နည်းတူစွာ `fc` ကို Run ကြည့်ပါ၊ ၎င်းက ဘာလုပ်ပေးသနည်း?

1. **Keyboard shortcut များ**

မကြာခဏ အသုံးပြုသည့် အခြား Application များကဲ့သို့ပင် Keyboard shortcut များနှင့် ရင်းနှီးအောင် ပြုလုပ်ထားခြင်းသည် အကျိုးရှိလှပါသည်။ အောက်ပါ အရာများကို ရိုက်ထည့်ကြည့်ပြီး ၎င်းတို့ မည်သို့ အလုပ်လုပ်သည်၊ မည်သည့် အခြေအနေများတွင် ၎င်းတို့အကြောင်း သိထားလျှင် အဆင်ပြေနိုင်မည်ကို ရှာဖွေကြည့်ပါ။ အချို့အတွက် အွန်လိုင်းတွင် ရှာဖွေခြင်းက ပိုမို လွယ်ကူနိုင်ပါသည်။ (`^X` ဆိုသည်မှာ `Ctrl+X` ကို နှိပ်ခြင်း ဖြစ်သည်ကို သတိရပါ)

- `^A`, `^E`
- `^R`
- `^L`
- `^C`, `^\` နှင့် `^D`
- `^U` နှင့် `^Y`

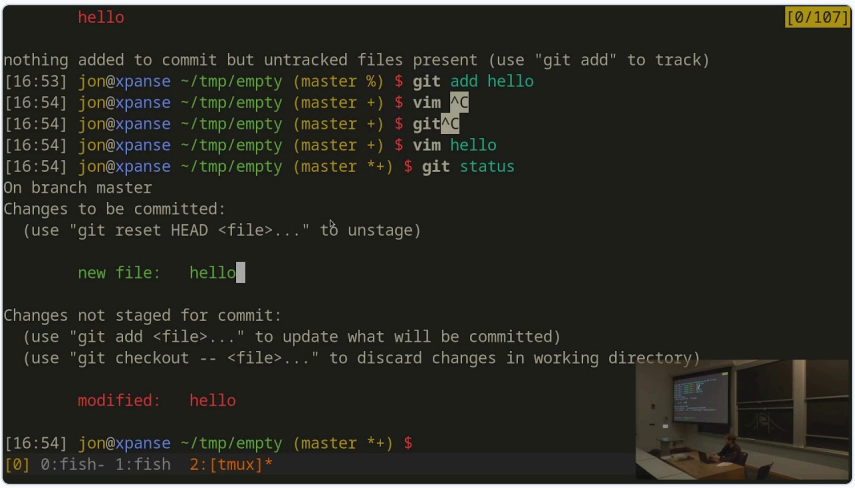
🔗 External Links & References (ပြင်ပ လင့်ခ်များနှင့် ကိုးကားချက်များ)

Resource / Description	URL	Micro QR
1. BashGuide	https://mywiki.wooledge.org/BashGuide	
2. The Linux Command Line	https://linuxcommand.org/tlcl.php	
3. shebang	https://en.wikipedia.org/wiki/Shebang_(Unix	
4. အားသာချက်၊ အားနည်းချက်များ	https://unix.stackexchange.com/questions/29608/why-is-it-better-to-use-usr-bin-env-name-instead-of-path-to-name-as-my	

Version Control

▶ LECTURE VIDEO REFERENCE

Version Control



Scan the QR code below using your mobile camera or click the link to watch this full lecture online:

Direct Online Link:

https://www.youtube.com/watch?v=3fig2Vz8QXs

အချိန်နှင့်အမျှ ပြောင်းလဲနေသည့် အရာတစ်ခုခုကို မိမိလုပ်ဆောင်နေသည့်အခါ ထိုအပြောင်းအလဲများကို *ခြေရာခံ (track)* နိုင်ခြင်းက အသုံးဝင်ပါသည်။ ဤသို့ပြုလုပ်ရခြင်းအတွက် အကြောင်းအရင်းများစွာ ရှိပါသည် - ၎င်းသည် မည်သည့်အရာများ ပြောင်းလဲသွားခဲ့သည်၊ မည်သို့ ပြန်ပြင်ရမည်၊ မည်သူ ပြောင်းလဲခဲ့သည်နှင့် မည်သည့်အတွက်ကြောင့် ပြောင်းလဲခဲ့ရသည် စသည်တို့ကို မှတ်တမ်းတစ်ခုအနေဖြင့် ပေးစွမ်းနိုင်သောကြောင့် ဖြစ်သည်။ Version control systems (VCS) များသည် သင့်အား ထိုစွမ်းဆောင်ရည်ကို ပေးစွမ်းပါသည်။ ၎င်းတို့သည် သင့်အား ပြောင်းလဲမှုကို ဖော်ပြသည့် စာတို (message) တစ်ခုနှင့်အတူ ဖိုင်အစုအဝေး

တစ်ခုသို့ အပြောင်းအလဲများကို *commit* လုပ်ခွင့်ပြုသလို၊ အတိတ်က မိမိပြုလုပ်ခဲ့သော အပြောင်းအလဲများကို ပြန်လည် ကြည့်ရှုခြင်းနှင့် ပြန်ပြင်ခြင်းတို့ကိုလည်း ပြုလုပ်နိုင်စေပါသည်။

VCS အများစုသည် သုံးစွဲသူ အများအပြားကြားတွင် commit ရာဇဝင် (history) ကို မျှဝေခြင်းအား အထောက်အပံ့ ပေးပါသည်။ ဤသည်မှာ အဆင်ပြေချောမွေ့သော ပူးပေါင်းဆောင်ရွက်မှုကို ဖြစ်စေပါသည် - ကျွန်ုပ် ပြုလုပ်ခဲ့သည့် အပြောင်းအလဲများကို သင့်အနေဖြင့် မြင်တွေ့နိုင်ပြီး၊ သင် ပြုလုပ်ခဲ့သည့် အပြောင်းအလဲများကိုလည်း ကျွန်ုပ် မြင်တွေ့နိုင်ပါသည်။ ထို့အပြင် VCS သည် *အပြောင်းအလဲများ* ကို ခြေရာခံထားသောကြောင့် ကျွန်ုပ်တို့၏ အပြောင်းအလဲများသည် သီးခြားစီဖြစ်နေသမျှ ကာလပတ်လုံး ၎င်းတို့ကို မည်သို့ပေါင်းစပ်ရမည်ကို (အမြဲတမ်း မဟုတ်သော်လည်း) မကြာခဏ အလိုအလျောက် သီးခြားခွဲခြား ပေါင်းစပ်ပေးနိုင်ပါသည်။

၎င်းတို့ အထောက်အပံ့ပေးသည့် အရာများ၊ အလုပ်လုပ်ပုံများနှင့် သင် ချိတ်ဆက်ဆောင်ရွက်ပုံများအပေါ် မူတည်၍ ကွဲပြားမှုများစွာရှိသော VCS များ [မြောက်မြားစွာ](https://en.wikipedia.org/wiki/Comparison_of_version-control_software) ရှိကြပါသည်။ ဤနေရာတွင် ကျွန်ုပ်တို့သည် အသုံးအများဆုံး စနစ်တစ်ခုဖြစ်သည့် **git** (https://git-scm.com/) ကို အဓိကထား လေ့လာသွားမည်ဖြစ်သော်လည်း၊ **Mercurial** (https://www.mercurial-scm.org/) ကိုလည်း လေ့လာကြည့်ပါရန် အကြံပြုလိုပါသည်။

ထိုသို့ ပြောကြားပြီးနောက် – အဓိက မှတ်စုတိုများဆီသို့ သွားကြရအောင်။

git ဆိုတာ မည်းမှောင်တဲ့ အမှောင်မှော်ပညာ ဖြစ်ပါသလား။

လုံးဝ မဟုတ်ပါဘူး.. မိမိအနေနဲ့ data model ကို နားလည်ဖို့ပဲ လိုအပ်ပါတယ်။ ကျွန်ုပ်တို့ အသေးစိတ်အချက်အချို့ကို ကျော်လွန်သွားမည်ဖြစ်သော်လည်း၊ ယေဘုယျအားဖြင့် ပြောရလျှင် git ရဲ့ *အဓိက (core)* အရာကတော့ commit ဖြစ်ပါတယ်။

- commit တိုင်းတွင် “revision hash” ဟုခေါ်သော သီးသန့် အမည် (unique name) တစ်ခုရှိပါသည် `998622294a6c520db718867354bf98348ae3c7e2` ကဲ့သို့သော ရှည်လျားသည့် hash တစ်ခုဖြစ်ပြီး ၎င်းကို မကြာခဏ အတိုကောက် ရှေ့ဆက်ဖြစ်သော `9986222` အဖြစ် သုံးလေ့ရှိသည်
- commit တွင် ရေးသားသူ (author) + commit စာတို (commit message) ပါဝင်သည်
- ထို့အပြင် *ancestor commits (ဘိုးဘေး commit များ)* ရဲ့ hash လည်း ပါဝင်သည် ပုံမှန်အားဖြင့် ယခင် commit ရဲ့ hash သာ ဖြစ်လေ့ရှိသည်
- commit သည် commit ရဲ့ ဘိုးဘေးမှ ယခု commit သို့ မည်သို့ ရောက်ရှိလာကြောင်း ဖော်ပြသည့် *diff* တစ်ခုကိုလည်း ကိုယ်စားပြုသည် (ဥပမာ- ဤဖိုင်ရှိ ဤစာကြောင်းကို ဖျက်ရန်၊ ဤဖိုင်သို့ ဤစာကြောင်းများ ထည့်ရန်၊ ထိုဖိုင်ကို အမည်ပြောင်းရန် စသည်ဖြင့်)
    - လက်တွေ့တွင် git သည် အပြောင်းအလဲ မတိုင်မီနှင့် ပြောင်းလဲပြီးနောက် အခြေအနေ အပြည့်အစုံကို သိမ်းဆည်းပါသည်

- ခဏခဏ ပြောင်းလဲနေသော ဖိုင်ကြီးများကို သိမ်းဆည်းရန် မသင့်တော်ပါ!

စတင်ချိန်တွင် *repository* (ယေဘုယျအားဖြင့်: git စီမံခန့်ခွဲသော folder) ၌ မည်သည့် အကြောင်းအရာမျှ မရှိသေးသလို၊ မည်သည့် commit မျှလည်း မရှိသေးပါ။ ၎င်းကို စတင် ပြင်ဆင်ကြည့်ရအောင် -

```
$ git init hackers
$ cd hackers
$ git status
```

၌နေရာတွင် ထွက်ပေါ်လာသော output သည် ကျွန်ုပ်တို့အတွက် ကောင်းမွန်သော စတင်မှတ် တစ်ခုကို ပေးပါသည်။ အသေးစိတ် လေ့လာကြည့်ပြီး အားလုံးကို နားလည်အောင် လုပ်ဆောင်ကြရအောင်။

ပထမဦးစွာ “On branch master”.

- hash များကို အချိန်တိုင်း မသုံးချင်ကြပါ။
- branch များသည် hash များကို ညွှန်ပြနေသည့် အမည်များ ဖြစ်ကြသည်။
- master သည် ပုံမှန်အားဖြင့် “နောက်ဆုံး” commit အတွက် ပေးထားသော အမည်ဖြစ်သည်။ commit အသစ်တစ်ခု ပြုလုပ်တိုင်း master အမည်သည် ထို commit အသစ်၏ hash ကို ညွှန်ပြသွားမည် ဖြစ်သည်။
- အထူးအမည် `HEAD` သည် “လက်ရှိ” အမည်ကို ညွှန်ပြသည်။
- မိမိကိုယ်ပိုင် အမည်များကိုလည်း `git branch` (သို့မဟုတ် `git tag`) ဖြင့် ဖန်တီးနိုင်သည် ထို အကြောင်းကို နောက်မှ ပြန်လာပါမည်

“No commits yet” ကို ကျော်လိုက်ပါမည်၊ အဘယ်ကြောင့်ဆိုသော် ၎င်း၏ အဓိပ္ပာယ်မှာ ရှင်းလင်းပြီးသား ဖြစ်သောကြောင့်ဖြစ်သည်။

ထို့နောက် “nothing to commit”.

- commit တိုင်းတွင် သင်ပြုလုပ်ခဲ့သော အပြောင်းအလဲ အားလုံး၏ diff တစ်ခု ပါဝင်သည်။ သို့သော် ထို diff ကို ပထမဦးစွာ မည်သို့ တည်ဆောက်ပါသနည်း။
- ယခင် commit ပြီးကတည်းက သင်ပြုလုပ်ခဲ့သော အပြောင်းအလဲ *အားလုံး* ကို အမြဲတမ်း commit လုပ်၍ ရနိုင်သည်
  - အချို့သော အခါများတွင် ၎င်းတို့အနက်မှ အချို့ကိုသာ commit လုပ်ချင်မည်ဖြစ်သည် (ဥပမာ- `TODO` များကို မပါစေချင်ဘဲ)

    - အချို့သော အခါများတွင် အပြောင်းအလဲများကို commit အများအပြား ခွဲထုတ်ပြီး တစ်ခုစီအတွက် သီးခြား commit message ပေးချင်မည်ဖြစ်သည်

- git သည် commit တစ်ခု တည်ဆောက်ရန် အပြောင်းအလဲများကို *stage* လုပ်ခွင့် ပေးသည်
    - ဖိုင်တစ်ခု သို့မဟုတ် ဖိုင်များ၏ အပြောင်းအလဲများကို `git add` ဖြင့် staged အပြောင်းအလဲများ အတွင်းသို့ ထည့်သွင်းပါ ```bash

- ဖိုင်တစ်ခုအတွင်းရှိ အပြောင်းအလဲအချို့ကိုသာ ထည့်ရန် `git add -p` ကို သုံးပါ
- argument မပါပါက `git add` သည် “သိရှိပြီး ဖိုင်အားလုံး” အပေါ် အလုပ်လုပ်ပါသည်

```
   - ဖိုင်တစ်ခုကို ဖျက်ပစ်ပြီး ၎င်း၏ ဖျက်ဆီးမှုကို stage လုပ်ရန် `git rm` ကို သုံးပါ
   - staged အပြောင်းအလဲများ အစုအဝေးကို အလွတ်ဖြစ်စေရန် `git reset` ကို သုံးပါ
```bash
 - ဤသည်မှာ မိမိ၏ ဖိုင်များကို ပြောင်းလဲခြင်း *မပြုလုပ်ပါ* ဆိုသည်ကို သတိပြုပါ!
   ၎င်းသည် commit အတွင်းတွင် မည်သည့် အပြောင်းအလဲမျှ ပါဝင်မည် မဟုတ်ကြောင်းသာ *ဆိုလိုခြင်း* ဖြစ်သည်
 - staged အပြောင်းအလဲ အချို့ကိုသာ ဖယ်ရှားရန်-
   `git reset FILE` သို့မဟုတ် `git reset -p`
```

- staged ဖြစ်နေသော အပြောင်းအလဲများကို `git diff --staged` ဖြင့် စစ်ဆေးပါ
- ကျန်ရှိနေသေးသော အပြောင်းအလဲများကို `git diff` ဖြင့် ကြည့်ပါ
- stage အခြေအနေကို ကျေနပ်ပါက `git commit` ဖြင့် commit တစ်ခု ဖန်တီးပါ

```
 - အပြောင်းအလဲ *အားလုံး* ကို သာ commit လုပ်လိုပါက: `git commit -a`
 - `git help add` တွင် အသုံးဝင်သော အချက်အလက်များစွာ ပါရှိသည်
```

အထက်ပါ အချက်များကို လေ့ကျင့်နေစဉ်အတွင်း git က သင့်ကို မည်သည့်အရာ ပြုလုပ်နေသည်ဟု ထင်မြင်နေကြောင်း ကြည့်ရှုရန် `git status` ကို စမ်းသပ် run ကြည့်ပါ – ၎င်းသည် မထင်မှတ်ထားလောက်အောင် အသုံးဝင်ပါသည်။

commit တစ်ခု ရပြီဆိုတော့...

ကဲ၊ ကျွန်ုပ်တို့မှာ commit တစ်ခု ရပါပြီ၊ အခု ဘာဆက်လုပ်ကြမလဲ။

- လတ်တလော အပြောင်းအလဲများကို ကြည့်ရှုနိုင်သည်: `git log` (သို့မဟုတ် `git log --oneline`)
- အပြောင်းအလဲ အပြည့်အစုံကို ကြည့်ရှုနိုင်သည်: `git log -p`
```

- သီးခြား commit တစ်ခုကို ပြသနိုင်သည်: `git show master`
  - သို့မဟုတ် အပြည့်အစုံ diff/patch အတွက် `-p` ဖြင့် ပြနိုင်သည်
- `git checkout NAME` ကို အသုံးပြု၍ commit တစ်ခု၏ အခြေအနေသို့ ပြန်သွားနိုင်သည်
  - အကယ်၍ `NAME` သည် commit hash တစ်ခုဖြစ်ပါက git က ကျွန်ုပ်တို့ကို “detached” ဖြစ်နေသည်ဟု ပြောပါလိမ့်မည်။ ဤသည်မှာ ထို commit ကို ညွှန်ပြသော `NAME` မရှိကြောင်းကိုသာ ဆိုလိုသဖြင့်၊ အကယ်၍ ကျွန်ုပ်တို့ commit များကို ပြုလုပ်ပါက မည်သူမျှ သိရှိနိုင်မည် မဟုတ်ပါ။
- `git revert NAME` ဖြင့် အပြောင်းအလဲတစ်ခုကို ပြန်လည် ပယ်ဖျက် (revert) နိုင်သည်
  - `NAME` ရှိ commit ၏ diff ကို ပြောင်းပြန်ပုံစံ ပြန်လည် သက်ရောက်စေပါသည်။
- `git diff NAME..` ကို အသုံးပြု၍ မူလ ဗားရှင်းအဟောင်းနှင့် ယခုဗားရှင်းကို နှိုင်းယှဉ်နိုင်သည်
  - `a..b` သည် commit *range (အပိုင်းအခြား)* ဖြစ်သည်။ တစ်ခုခုကို ချန်လှပ်ထားခဲ့ပါက ၎င်းသည် `HEAD` ကို ဆိုလိုသည်။
- `git log NAME..` ကို အသုံးပြု၍ ကြားရှိ commit များအားလုံးကို ပြသနိုင်သည်
  - ဤနေရာတွင်လည်း `-p` သည် အလုပ်လုပ်ပါသည်
- `git reset NAME` ဖြင့် သီးခြား commit တစ်ခုကို ညွှန်ပြရန် `master` ကို ပြောင်းလဲနိုင်သည် (ထိုအချိန်မှစ၍ ပြုလုပ်ခဲ့သမျှ အရာအားလုံးကို ထိရောက်စွာ ပြန်ဖျက်ပစ်ခြင်းဖြစ်သည်):
  - ဩော်၊ ဘာလို့ပါလဲ။ `reset` က staged အပြောင်းအလဲတွေကို ပြောင်းလဲဖို့ မဟုတ်ဘူးလား။

    ```
    reset တွင် "ဒုတိယ" ပုံစံရှိပါသည် (`git help reset` ကို ကြည့်ပါ)၊ ၎င်းသည် ပေးထားသော အမည်ဖြင့် ညွှန်ပြထားသော commit သို့ `HEAD` ကို သတ်မှတ်ပေးပါသည်။
    ```

  - ဤသည်မှာ မည်သည့် ဖိုင်ကိုမျှ မပြောင်းလဲခဲ့ကြောင်း သတိပြုပါ – ယခု `git diff` သည် `git diff NAME..` ကို ထိရောက်စွာ ပြသပေးနေမည်ဖြစ်ပါသည်။

အမည်တစ်ခုမှာ ဘာတွေ ပါဝင်နေသလဲ။

ရှင်းရှင်းလင်းလင်းပင် git တွင် အမည်များသည် အရေးပါပါသည်။ ၎င်းတို့သည် git အတွင်း ဖြစ်ပျက်နေသည် များ၏ *အရာများစွာ* ကို နားလည်ရန် သော့ချက်ဖြစ်ကြသည်။ ယခုအချိန်အထိ ကျွန်ုပ်တို့သည် commit hash များ၊ master နှင့် `HEAD` တို့အကြောင်း ပြောဆိုခဲ့ပြီး ဖြစ်သည်။ သို့သော် နောက်ထပ် ရှိပါသေးသည်!

- `git branch b` ဖြင့် မိမိကိုယ်ပိုင် branch များကို (master ကဲ့သို့) ဖန်တီးနိုင်သည်

    - `HEAD` ရှိ commit ကို ညွှန်ပြသော အမည်အသစ် `b` ကို ဖန်တီးပေးပါသည်
    - သို့သော် သင်သည် master ပေါ်တွင် ရှိနေဆဲဖြစ်သဖြင့် commit အသစ်တစ်ခု ပြုလုပ်ပါက master သည် ထို commit အသစ်ကို ညွှန်ပြမည်ဖြစ်ပြီး `b` က ညွှန်ပြမည် မဟုတ်ပါ။
    - `git checkout b` ဖြင့် branch တစ်ခုသို့ ပြောင်းပါ ```bash

- သင်ပြုလုပ်သမျှ commit များသည် ယခုအခါ `b` အမည်ကို update ပြုလုပ်ပါလိမ့်မည်
- `git checkout master` ဖြင့် master သို့ ပြန်ပြောင်းပါ
    - `b` အတွင်းရှိ သင့်အပြောင်းအလဲများ အားလုံးကို ကွယ်ဝှက်ထားရှိပါသည်
- အပြောင်းအလဲများကို လွယ်ကူစွာ စမ်းသပ်နိုင်သည့် လွန်စွာ အသုံးဝင်သော နည်းလမ်းဖြစ်သည်

```
 - tag များသည် မည်သည့်အခါမျှ မပြောင်းလဲဘဲ သီးခြား စာတိုပါရှိသော အခြား အမည်များ ဖြစ်ကြသည်။
၎င်းတို့ကို release များ + changelog များကို မှတ်သားရန် မကြာခဏ အသုံးပြုကြသည်။
 - `NAME^` သည် "`NAME` ၏ ရှေ့မှ commit" ကို ဆိုလိုသည်
   - ထပ်ခါထပ်ခါ အသုံးပြုနိုင်သည်: `NAME^^^`
   - သင် `~` ကို သုံးသည့်အခါ _အများအားဖြင့်_ `~` ကို ဆိုလိုခြင်း ဖြစ်နိုင်သည်
```bash
 - `~` သည် "အချိန်ကာလအလိုက်" ဖြစ်ပြီး၊ `^` သည် ဘိုးဘေးစဉ်ဆက်အတိုင်း သွားပါသည်
 - `~~` သည် `^^` နှင့် အတူတူပင်ဖြစ်သည်
 - `~` ဖြင့် `X` ထက် "၃ စောင်မက စောသော commit" အတွက် `X~3` ဟုလည်း ရေးနိုင်သည်
 - သင် `^3` ကို လိုချင်မည် မဟုတ်ပါ
```

- `git diff HEAD^`
- `-` သည် “ယခင် အမည်” ကို ဆိုလိုသည်
- အခြား argument ကို မပေးပါက မိန့်ခွန်းအများစုသည် `HEAD` အပေါ် အလုပ်လုပ်ပါသည်

သင်၏ ရှုပ်ပွနေသည်များကို ရှင်းလင်းပါ

သင်၏ commit ရာဇဝင် (history) သည် *မကြာခဏ* အော်ပါအတိုင်း ဖြစ်သွားလေ့ရှိပါသည် -

- `add feature x` – `x` အကြောင်း commit message တစ်ခု ပါနိုင်သည်!
- `forgot to add file`
- `fix bug`
- `typo`
- `typo2`
```

- `actually fix`
- `actually actually fix`
- `tests pass`
- `fix example code`
- `typo`
- `x`
- `x`
- `x`
- `x`

git ၏ ရှုထောင့်မှကြည့်လျှင် ၎င်းမှာ *အဆင်ပြေ* သော်လည်း၊ အနာဂတ် သင်ကိုယ်တိုင်အတွက် သို့မဟုတ် မည်သည့်အရာများ ပြောင်းလဲသွားသည်ကို သိရှိလိုသော အခြားသူများအတွက် မည်သို့မျှ အထောက်အကူ မပြုပါ။ git သည် သင့်အား ဤအရာများကို ရှင်းလင်းခွင့် ပြုထားသည် -

- `git commit --amend` : staged ဖြစ်နေသော အပြောင်းအလဲများကို ယခင် commit အတွင်းသို့ ပေါင်းထည့်ပါ
  - ဤသည်မှာ ယခင် commit ကို *ပြောင်းလဲစေပြီး* hash အသစ်တစ်ခု ပေးသွားမည်ဖြစ်ကြောင်း သတိပြုပါ!
- `git rebase -i HEAD~13` သည် *အလွန်ပင် ဆန်းကြယ်ပါသည်*။ လွန်ခဲ့သော 13 စောင်မှ commit တစ်ခုစီအတွက် မည်သို့ ပြုလုပ်ရမည်ကို ရွေးချယ်ပါ-
  - ပုံမှန်အားဖြင့် `pick` ဖြစ်သည်; မည်သည့်အရာမျှ မပြုလုပ်ပါ
  - `r` : commit message ကို ပြောင်းလဲရန်
  - `e` : commit ကို ပြောင်းလဲရန် (ဖိုင်များ ထည့်ရန် သို့မဟုတ် ဖျက်ရန်)
  - `s` : commit ကို ယခင် commit နှင့် ပေါင်းစပ်ပြီး commit message ကို ပြင်ဆင်ရန်
  - `f` : “fixup” – commit ကို ယခင် commit နှင့် ပေါင်းစပ်ရန်; commit message ကို ပယ်ဖျက်ရန်
  - ပြီးဆုံးပါက `HEAD` ကို ယခုအခါ နောက်ဆုံး commit ဖြစ်နေသော အရာဆီသို့ ညွှန်ပြစေသည်
  - ၎င်းကို commit များကို *squash ပြုလုပ်ခြင်း* ဟု မကြာခဏ ခေါ်ဆိုလေ့ရှိသည်
  - ၎င်းအမှန်တကယ် ပြုလုပ်သည့်အရာ- rebase စတင်သည့် အမှတ်သို့ `HEAD` ကို ပြန်ရစ်ပြီး၊ ထို့နောက် လမ်းညွှန်ချက်အတိုင်း commit များကို အစဉ်လိုက် ပြန်လည် အသုံးပြုခြင်းဖြစ်သည်။

- `git reset --hard NAME` : ဖိုင်အားလုံး၏ အခြေအနေကို `NAME` (သို့မဟုတ် အမည်မပေးထားပါက `HEAD`) ၏ အခြေအနေသို့ ပြန်လည် reset လုပ်ပါ။ အပြောင်းအလဲများကို ပြန်ဖြုတ်ရန် လွန်စွာ အသုံးဝင်ပါသည်။

အခြားသူများနှင့် ပူးပေါင်းဆောင်ရွက်ခြင်း

version control ၏ အသုံးများသော သုံးစွဲပုံတစ်ခုမှာ လူအများအပြားအား တစ်ဦးနှင့်တစ်ဦး အနှောင့်အယှက်မဖြစ်စေဘဲ ဖိုင်အစုအဝေးတစ်ခုကို ပြောင်းလဲမှုများ ပြုလုပ်ခွင့် ပေးခြင်းဖြစ်ပါသည်။ သို့မဟုတ် အကယ်၍ တစ်ဦးနှင့်တစ်ဦး အပြောင်းအလဲများ ထပ်သွားပါကလည်း၊ အခြားသူ၏ အပြောင်းအလဲများကို တိတ်တဆိတ် ထပ်ရေးမိခြင်း (overwrite) မဖြစ်စေရန် သေချာစေခြင်း ဖြစ်ပါသည်။

git သည် *distributed (ဗဟိုချုပ်ကိုင်မှုမရှိသော)* VCS ဖြစ်ပါသည် - လူတိုင်းတွင် တစ်ခုလုံးသော repository ၏ local copy တစ်ခုစီ (အခြားသူများ ထုတ်ဝေရန် ရွေးချယ်ထားသော အရာအားလုံး) ရှိကြသည်။ အချို့သော VCS များမှာ *centralized (ဗဟိုချုပ်ကိုင်သော)* စနစ်များ ဖြစ်ကြသည် (ဥပမာ- subversion) - server တွင် commit အားလုံး ရှိပြီး client များတွင် ၎င်းတို့ “checkout” လုပ်ထားသော ဖိုင်များသာ ရှိကြသည်။ အခြေခံအားဖြင့် ၎င်းတို့တွင် *လက်ရှိ* ဖိုင်များသာ ရှိပြီး အခြားအရာတစ်ခုခု လိုအပ်ပါက server ကို တောင်းဆိုရန် လိုအပ်သည်။

git repository တစ်ခု၏ copy တိုင်းကို “remote” တစ်ခုအဖြစ် စာရင်းသွင်းနိုင်ပါသည်။ `git clone ADDRESS` ကို အသုံးပြု၍ ရှိပြီးသား git repository တစ်ခုကို မိတ္တူကူးယူနိုင်ပါသည် (`git init` အစား)။ ဤသည်မှာ `ADDRESS` ကို ညွှန်ပြသော *origin* ဟုခေါ်သော remote တစ်ခုကို ဖန်တီးပေးပါသည်။ remote တစ်ခုမှ အမည်များနှင့် ၎င်းတို့ ညွှန်ပြသော commit များကို `git fetch REMOTE` ဖြင့် ရယူနိုင်ပါသည်။ remote တစ်ခုရှိ အမည်များ အားလုံးကို သင့်ထံတွင် `REMOTE/NAME` အဖြစ် ရရှိနိုင်ပြီး၊ ၎င်းတို့ကို local အမည်များကဲ့သို့ပင် အသုံးပြုနိုင်ပါသည်။

အကယ်၍ သင့်တွင် remote သို့ ရေးသားခွင့် (write access) ရှိပါက၊ `git push` ကို အသုံးပြု၍ သင်ပြုလုပ်ခဲ့သော commit များကို ညွှန်ပြရန် remote ရှိ အမည်များကို ပြောင်းလဲနိုင်ပါသည်။ ဥပမာအားဖြင့် `origin` အမည်ရှိ remote ၌ master အမည် (branch) ကို ကျွန်ုပ်တို့၏ master branch မှ လက်ရှိ ညွှန်ပြနေသော commit သို့ ညွှန်ပြစေကြစို့ -

- `git push origin master:master`
- အဆင်ပြေစေရန်အတွက် လက်ရှိ branch မှ `git push` လုပ်သည့်အခါ မူလသတ်မှတ်ထားသော ပစ်မှတ်အဖြစ် `origin/master` ကို `-u` ဖြင့် သတ်မှတ်နိုင်ပါသည်
- စဉ်းစားကြည့်ပါ- ဤမိန့်ခွန်းသည် မည်သည့်အရာကို ပြုလုပ်သနည်း။ `git push origin master:HEAD^`

မကြာခဏဆိုသလို သင်သည် သင်၏ remote အဖြစ် GitHub၊ GitLab၊ BitBucket သို့မဟုတ် အခြားတစ်ခုခုကို အသုံးပြုမည် ဖြစ်သည်။ git ၏ ရှုထောင့်မှကြည့်လျှင် ထိုအရာများအတွက် "ထူးခြားမှု" မရှိပါ။ အရာအားလုံးသည် အမည်များနှင့် commit များသာ ဖြစ်ကြသည်။ အကယ်၍ တစ်စုံတစ်ယောက်က master သို့ အပြောင်းအလဲတစ်ခု ပြုလုပ်ပြီး ၎င်းတို့၏ commit ကို ညွှန်ပြရန် `github/master` ကို update လုပ်လိုက်ပါက (ထိုအကြောင်းကို ခဏအတွင်း ပြန်လာပါမည်)၊ သင် `git fetch github` ပြုလုပ်သည့်အခါ ၎င်းတို့၏ အပြောင်းအလဲများကို `git log github/master` ဖြင့် ကြည့်ရှုနိုင်မည် ဖြစ်သည်။

အခြားသူများနှင့် လက်တွဲလုပ်ဆောင်ခြင်း

ယခုအချိန်အထိ branch များသည် သိပ်မသုံးဝင်သကဲ့သို့ ထင်ရပါသည်- ၎င်းတို့ကို ဖန်တီးနိုင်သည်၊ ၎င်းတို့ထဲတွင် အလုပ်လုပ်နိုင်သည်၊ သို့သော် ထို့နောက်တွင်ကော။ နောက်ဆုံးတွင် သင်သည် master ကို ၎င်းတို့ထံ ညွှန်ပြအောင် လုပ်ဆောင်မည် မဟုတ်ပါလား။

- လုပ်ဆောင်ချက်ကြီး တစ်ခု (big feature) ပေါ်တွင် အလုပ်လုပ်နေစဉ် တစ်ခုခုကို ပြင်ဆင်ရန် လိုအပ်လာပါက မည်သို့ ပြုလုပ်မည်နည်း။
- ထိုအတောအတွင်း အခြားတစ်စုံတစ်ယောက်က master သို့ အပြောင်းအလဲတစ်ခု ပြုလုပ်လိုက်ပါက မည်သို့ ပြုလုပ်မည်နည်း။

ရှောင်လွှဲ၍ မရနိုင်စွာပင် branch တစ်ခုရှိ အပြောင်းအလဲများကို အခြား branch တစ်ခုရှိ အပြောင်းအလဲများနှင့် *merge (ပေါင်းစပ်)* ရမည် ဖြစ်သည်၊ ထို အပြောင်းအလဲများကို သင်ကိုယ်တိုင် သို့မဟုတ် အခြားသူက ပြုလုပ်ခဲ့သည်ဖြစ်စေ။ git က သင့်အား ၎င်းကို မအံ့သြစရာပင် `git merge NAME` ဖြင့် ပြုလုပ်ခွင့် ပေးထားသည်။ `merge` က ဆောင်ရွက်မည့် အရာများမှာ -

- `HEAD` နှင့် `NAME` တို့သည် ဘိုးဘေး commit အတူတူ မျှဝေခဲ့သည့် (ဆိုလိုသည်မှာ ၎င်းတို့ ခွဲထွက်ခဲ့သည့်) နောက်ဆုံးအမှတ်ကို ရှာဖွေခြင်း
- ထို အပြောင်းအလဲများ အားလုံးကို လက်ရှိ `HEAD` သို့ သက်ရောက်စေရန် (ကြိုးစားခြင်း)
- ထို အပြောင်းအလဲများ အားလုံး ပါဝင်သော commit တစ်ခုကို ထုတ်လုပ်ပြီး `HEAD` နှင့် `NAME` နှစ်ခုလုံးကို ၎င်း၏ ဘိုးဘေးများအဖြစ် စာရင်းသွင်းခြင်း
- `HEAD` ကို ထို commit ၏ hash အဖြစ် သတ်မှတ်ခြင်း

သင်၏ လုပ်ဆောင်ချက်ကြီး ပြီးစီးပါက ၎င်း၏ branch ကို master အတွင်းသို့ merge လုပ်နိုင်ပြီး၊ git က မည်သည့် branch မှ အပြောင်းအလဲ မည်သည့်အရာမျှ ဆုံးရှုံးမသွားစေရန် သေချာအောင် ပြုလုပ်ပေးပါလိမ့်မည်။

အကယ်၍ သင်သည် အတိတ်က git ကို အသုံးပြုခဲ့ဖူးပါက `merge` ကို အခြားအမည်တစ်ခု ဖြစ်သည့် `pull` အဖြစ် သတိပြုမိနိုင်ပါသည်။ `git pull REMOTE BRANCH` ကို ပြုလုပ်သည့်အခါ ၎င်းသည် -

- `git fetch REMOTE`
- `git merge REMOTE/BRANCH`
- ဤနေရာတွင် `push` ကဲ့သို့ပင် `REMOTE` နှင့် `BRANCH` တို့ကို မကြာခဏ ချန်လှပ်ထားလေ့ရှိပြီး “tracking” ဖြစ်သော remote branch ကို အသုံးပြုကြသည် (`-u` ကို မှတ်မိပါသလား)

merge ပြုလုပ်မည့် branch များ၏ အပြောင်းအလဲများသည် သီးခြားစီ ဖြစ်နေသမျှ ကာလပတ်လုံး ဤသည်မှာ ပုံမှန်အားဖြင့် *အလွန် ကောင်းမွန်စွာ* အလုပ်လုပ်ပါသည်။ အကယ်၍ ၎င်းတို့ သီးခြားစီ မဟုတ်ပါက သင့်ထံတွင် *merge conflict (ပေါင်းစပ်မှု ပဋိပက္ခ)* ဖြစ်ပေါ်လာမည် ဖြစ်သည်။ ခြောက်ခြားဖွယ် ကြားရသော်လည်း...

- merge conflict ဆိုသည်မှာ နောက်ဆုံး diff က မည်သို့ပုံစံ ဖြစ်သင့်သည်ကို မသိရှိကြောင်း git က သင့်အား ပြောပြခြင်းသာ ဖြစ်သည်
- git က ခေတ္တရပ်တန့်ပြီး “merge commit” ကို stage လုပ်ခြင်း ပြီးစီးအောင် ပြုလုပ်ရန် တောင်းဆိုသည်
- ပဋိပက္ခဖြစ်နေသော ဖိုင်ကို သင်၏ editor တွင် ဖွင့်ပြီး ထောင့်ကွင်း အများအပြား (`<<<<<<<`) ကို ရှာဖွေပါ။ `=======` ၏ အထက်ရှိ အရာများသည် မျှဝေထားသော ဘိုးဘေး commit ပြီးကတည်းက `HEAD` တွင် ပြုလုပ်ခဲ့သော အပြောင်းအလဲ ဖြစ်သည်။ အောက်ရှိ အရာများသည် မျှဝေထားသော commit ပြီးကတည်းက `NAME` တွင် ပြုလုပ်ခဲ့သော အပြောင်းအလဲ ဖြစ်သည်။
- `git mergetool` သည် အလွန် အသုံးဝင်ပါသည် – diff editor တစ်ခုကို ဖွင့်ပေးသည်
- ဖိုင်သည် ယခု မည်သို့ဖြစ်သင့်သည်ကို အဖြေရှာခြင်းဖြင့် ပဋိပက္ခကို *ဖြေရှင်း (resolve)* ပြီးပါက ထို အပြောင်းအလဲများကို `git add` ဖြင့် stage လုပ်ပါ
- ပဋိပက္ခ အားလုံးကို ဖြေရှင်းပြီးပါက `git commit` ဖြင့် ပြီးဆုံးအောင် ဆောင်ရွက်ပါ
  - `git merge --abort` ဖြင့် လက်လျှော့ စွန့်လွှတ်နိုင်ပါသည်

သင်သည် သင်၏ ပထမဆုံး git merge conflict ကို ဖြေရှင်းပြီးပါပြီ။ / ယခုအခါ ပြီးစီးသွားသော အပြောင်းအလဲများကို `git push` ဖြင့် ထုတ်ဝေနိုင်ပါပြီ။

ကမ္ဘာများ ထိတွေ့မိသည့်အခါ

သင် `push` ပြုလုပ်သည့်အခါ၊ သင် push လုပ်နေသည့် remote အမည်ကို update ပြုလုပ်ပါက အခြားသူ၏ အလုပ်များ ဆုံးရှုံးမသွားစေရန် git က စစ်ဆေးပါသည်။ ၎င်းသည် remote အမည်၏ လက်ရှိ commit သည် သင် push လုပ်နေသော commit ၏ ဘိုးဘေးဖြစ်မဖြစ် စစ်ဆေးခြင်းဖြင့် ပြုလုပ်သည်။ အကယ်၍ ဖြစ်ပါက

git သည် အမည်ကို ဘေးကင်းစွာ update ပြုလုပ်နိုင်သည်၊ ၎င်းကို *fast-forwarding* ဟု ခေါ်သည်။ အကယ်၍ မဟုတ်ပါက git သည် remote အမည်ကို update ပြုလုပ်ရန် ငြင်းဆန်မည်ဖြစ်ပြီး အပြောင်းအလဲများ ရှိနေကြောင်း သင့်အား ပြောပြပါလိမ့်မည်။

အကယ်၍ သင်၏ push ကို ငြင်းဆန်ခံရပါက သင် မည်သို့ ပြုလုပ်မည်နည်း။

- `git pull` ဖြင့် remote အပြောင်းအလဲများကို merge ပြုလုပ်ပါ (ဆိုလိုသည်မှာ `fetch` + `merge`)
- `--force` ဖြင့် push ကို အတင်းအကျပ် ပြုလုပ်ပါ: ဤသည်မှာ အခြားသူများ၏ အပြောင်းအလဲများကို ဆုံးရှုံးစေမည် ဖြစ်သည်!
  - `--force-with-lease` လည်း ရှိပါသေးသည်၊ ၎င်းသည် ထို remote မှ သင်နောက်ဆုံး အကြိမ် fetch ပြုလုပ်ခဲ့ပြီးနောက်ပိုင်း remote အမည် ပြောင်းလဲမသွားပါက အပြောင်းအလဲကို အတင်းအကျပ် ပြုလုပ်မည်ဖြစ်သည်။ များစွာ ပိုမိုဘေးကင်းပါသည်!
  - အကယ်၍ သင်သည် ယခင်က push ခဲ့ဖူးသော local commit များကို rebase ပြုလုပ်ခဲ့ပါက (“history ရေးသားပြင်ဆင်ခြင်း”၊ ၎င်းကို မပြုလုပ်တာ ပိုကောင်းနိုင်သည်)၊ သင် force push ပြုလုပ်ရလိမ့်မည်။ အဘယ်ကြောင့်ဆိုသည်ကို စဉ်းစားကြည့်ပါ!
- remote တွင် ပြုလုပ်ခဲ့သော အပြောင်းအလဲများ၏ “အပေါ်၌” သင်၏ အပြောင်းအလဲများကို ပြန်လည် သက်ရောက်စေရန် ကြိုးစားပါ
  - ဤသည်မှာ `rebase` ဖြစ်ပါသည်! ```bash
- မျှဝေထားသော ဘိုးဘေး ပြီးကတည်းက local commit အားလုံးကို ပြန်ရစ်ပါ
- `HEAD` ကို remote အမည်ရှိ commit ထံ fast-forward ပြုလုပ်ပါ
- local commit များကို အစဉ်လိုက် ပြန်လည် သက်ရောက်စေပါ
  - သင့်အနေဖြင့် ကိုယ်တိုင် ဖြေရှင်းရမည့် conflict များ ရှိကောင်းရှိနိုင်သည်
  - `git rebase --continue` သို့မဟုတ် `--abort`
  - အသေးစိတ်ကို ဤနေရာတွင် (https://git-scm.com/book/en/v2/Git-Branching-Rebasing) ထပ်မံ ကြည့်ရှုပါ ```
  - `git pull --rebase` သည် သင့်အတွက် ဤလုပ်ငန်းစဉ်ကို စတင်ပေးမည် ဖြစ်သည်
  - သင် merge သို့မဟုတ် rebase ပြုလုပ်သင့်သလား ဆိုသည်မှာ အပူတပြင်း ဆွေးနွေးနေကြသော ခေါင်းစဉ်တစ်ခု ဖြစ်သည်! အောက်ပါတို့မှာ ဖတ်ရှုရန် ကောင်းမွန်သော ဆောင်းပါးများ ဖြစ်ကြသည်- ```bash
- ဤဆောင်းပါး (https://www.atlassian.com/git/tutorials/merging-vs-rebasing)

- ဤဆောင်းပါး (https://web.archive.org/web/20210106220723/https://derekgourlay.com/blog/git-when-to-merge-vs-when-to-rebase/)
- ဤဆောင်းပါး (https://stackoverflow.com/questions/804115/when-do-you-use-git-rebase-instead-of-git-merge)

```
# ထပ်မံ ဖတ်ရှုရန်များ

<a href="https://imgs.xkcd.com/comics/git.png" class="ext-link">![git အကြောင်း X
KCD</a> <span class="ext-url-print">(https://imgs.xkcd.com/comics/git.png)</spa
n>](https://xkcd.com/1597/)

 - <a href="https://learngitbranching.js.org/" class="ext-link">Learn git branc
hing</a> <span class="ext-url-print">(https://learngitbranching.js.org/)</span>
 - <a href="https://smusamashah.github.io/blog/2017/10/14/explain-git-in-simple
-words" class="ext-link">How to explain git in simple words</a> <span class="ex
t-url-print">(https://smusamashah.github.io/blog/2017/10/14/explain-git-in-simp
le-words)</span>
 - <a href="https://jwiegley.github.io/git-from-the-bottom-up/" class="ext-lin
k">Git from the bottom up</a> <span class="ext-url-print">(https://jwiegley.git
hub.io/git-from-the-bottom-up/)</span>
 - <a href="https://eagain.net/articles/git-for-computer-scientists/" class="ex
t-link">Git for computer scientists</a> <span class="ext-url-print">(https://ea
gain.net/articles/git-for-computer-scientists/)</span>
 - <a href="https://ohshitgit.com/" class="ext-link">Oh shit, git!</a> <span cl
ass="ext-url-print">(https://ohshitgit.com/)</span>
 - <a href="https://git-scm.com/book/en/v2" class="ext-link">The Pro Git book</
a> <span class="ext-url-print">(https://git-scm.com/book/en/v2)</span>

# လေ့ကျင့်ခန်းများ

1. repository တစ်ခုတွင် ရှိပြီးသား ဖိုင်တစ်ခုကို ပြင်ဆင်ကြည့်ပါ။ `git stash` ကို ပြုလုပ်သည့်အခါ
မည်သို့ ဖြစ်ပျက်သနည်း။ `git log --all --oneline` ကို run သည့်အခါ သင့်အနေဖြင့် မည်သည့်အ
ရာကို မြင်တွေ့ရသနည်း။ `git stash` ဖြင့် ပြုလုပ်ခဲ့သည်များကို ပြန်ပြင်ရန် `git stash pop` ကို run
ပါ။ မည်သည့် အခြေအနေမျိုးတွင် ဤသည်မှာ အသုံးဝင်နိုင်သနည်း။

1. git ကို လေ့လာရာတွင် တွေ့ရလေ့ရှိသော အမှားတစ်ခုမှာ git ဖြင့် မစီမံသင့်သော ဖိုင်ကြီးများကို commi
t လုပ်မိခြင်း သို့မဟုတ် ထိခိုက်လွယ်သော အချက်အလက်များ ထည့်သွင်းမိခြင်း ဖြစ်သည်။ repository သို့
ဖိုင်တစ်ခု ထည့်သွင်းကြည့်ပါ၊ commit အချို့ ပြုလုပ်ပါ၊ ထို့နောက် ထိုဖိုင်ကို ရာဇဝင် (history) မှ ဖျက်ပ
စ်ပါ (<a href="https://help.github.com/articles/removing-sensitive-data-from-a-re
pository/" class="ext-link">ဤဆောင်းပါး</a> <span class="ext-url-print">(http
s://help.github.com/articles/removing-sensitive-data-from-a-repository/)</span>
ကို လေ့လာလိုပါလိမ့်မည်)။ ထို့အပြင် git အား သင့်အတွက် ဖိုင်ကြီးများကို စီမံပေးစေလိုပါက <a href
="https://git-lfs.github.com/" class="ext-link">Git-LFS</a> <span class="ext-ur
l-print">(https://git-lfs.github.com/)</span> ကို လေ့လာကြည့်ပါ။

1. Git သည် အပြောင်းအလဲများကို ပြန်ဖြုတ်ရန်အတွက် အမှန်တကယ် အဆင်ပြေလှသော်လည်း ဖြစ်တောင့်
ဖြစ်ခဲ အပြောင်းအလဲများနှင့်ပင် ကျွမ်းဝင်မှု ရှိရန် လိုအပ်သည်
   1. အကယ်၍ ဖိုင်တစ်ခုကို commit တစ်ခုခုတွင် မှားယွင်း ပြောင်းလဲခဲ့ပါက ၎င်းကို `git revert` ဖြ
င့် ပြန်လည် ပယ်ဖျက်နိုင်ပါသည်။ သို့သော် commit တစ်ခုတွင် အပြောင်းအလဲ အများအပြား ပါဝင်နေပါက `
revert` သည် အကောင်းဆုံး ရွေးချယ်မှု မဟုတ်နိုင်ပါ။ သီးခြား commit တစ်ခုမှ ဖိုင်ဗားရှင်းကို ပြန်လည်
```

ရယူရန် `git checkout` ကို မည်သို့ အသုံးပြုနိုင်သနည်း။
    1. branch တစ်ခု ဖန်တီးပါ၊ ထို branch ၌ commit တစ်ခု ပြုလုပ်ပါ၊ ထို့နောက် ၎င်းကို ဖျက်ပစ်ပါ။ ထို commit ကို ပြန်လည် ရယူနိုင်ပါသေးသလား။ `git reflog` ကို လေ့လာကြည့်ပါ။ (မှတ်ချက်- တွဲလောင်းဖြစ်နေသော အရာများကို လျင်မြန်စွာ ပြန်လည်ရယူပါ၊ မည်သည့်အရာကမျှ ညွှန်ပြမနေသော commit များကို git က ပုံမှန် အလိုအလျောက် ရှင်းလင်းပေးပါလိမ့်မည်။)
    1. အကယ်၍ `git reset` အစား `git reset --hard` ကို အလျင်စလို အသုံးပြုပါက အပြောင်းအလဲများ လွယ်ကူစွာ ဆုံးရှုံးသွားနိုင်သည်။ သို့သော် အပြောင်းအလဲများမှာ staged ဖြစ်ခဲ့ပြီး ဖြစ်သောကြောင့် ၎င်းတို့ကို ကျွန်ုပ်တို့ ပြန်လည် ရယူနိုင်ပါသည်။ (`git fsck --lost-found` နှင့် `.git/lost-found` တို့ကို လေ့လာကြည့်ပါ)

1. မည်သည့် git repo မဆို `.git/hooks` folder အောက်တွင် ကြည့်ပါ၊ `.sample` ဖြင့် ဆုံးသော script များစွာကို တွေ့ရပါလိမ့်မည်။ အကယ်၍ ၎င်းတို့ကို `.sample` မပါဘဲ အမည်ပြောင်းလိုက်ပါက ၎င်းတို့၏ အမည်အပေါ် မူတည်၍ run မည်ဖြစ်သည်၊ ဥပမာအားဖြင့် `pre-commit` သည် commit မပြုလုပ်မီ execute လုပ်ပါလိမ့်မည်။ ၎င်းတို့ဖြင့် စမ်းသပ်ကြည့်ပါ

1. command line tool အများအပြားကဲ့သို့ပင် `git` သည် `~/.gitconfig` ဟုခေါ်သော configuration file (သို့မဟုတ် dotfile) တစ်ခုကို ပေးထားပါသည်။ `~/.gitconfig` ကို အသုံးပြု၍ alias တစ်ခု ဖန်တီးပါ၊ ထို့ကြောင့် သင် `git graph` ကို run သည့်အခါ `git log --oneline --decorate --all --graph` ၏ output ကို ရရှိမည် ဖြစ်သည် (ဤသည်မှာ commit graph ကို အမြန် ကြည့်ရှုရန် ကောင်းမွန်သော မိန့်ခွန်းဖြစ်သည်)

1. Git သည် သင့်အား `~/.gitignore_global` အောက်တွင် global ignore pattern များကို သတ်မှတ်ခွင့် ပြုထားသည်၊ ဤသည်မှာ RSA key များကို ထည့်မိခြင်းကဲ့သို့သော ပုံမှန် အမှားများကို ကာကွယ်ရန် အသုံးဝင်သည်။ `~/.gitignore_global` ဖိုင်တစ်ခု ဖန်တီးပြီး pattern `*rsa` ကို ထည့်ပါ၊ ထို့နောက် repo တစ်ခုတွင် အလုပ်လုပ် မလုပ် စမ်းသပ်ပါ။

1. သင် `git` နှင့် ပိုမို ကျွမ်းဝင်လာပါက၊ သင်၏ `.gitignore` ကို ပြင်ဆင်ခြင်းကဲ့သို့သော ပုံမှန် အလုပ်များကို ပြုလုပ်လာရသည်ကို တွေ့ရပါလိမ့်မည်။ git extras (https://github.com/tj/git-extras/blob/master/Commands.md) သည် `git` နှင့် တွဲဖက် အလုပ်လုပ်သော သေးငယ်သည့် utility များကို ထောက်ပံ့ပေးသည်။ ဥပမာအားဖြင့် `git ignore PATTERN` သည် သင်၏ repo ရှိ `.gitignore` ဖိုင်သို့ သတ်မှတ်ထားသော pattern ကို ထည့်သွင်းပေးမည်ဖြစ်ပြီး `git ignore-io LANGUAGE` သည် gitignore.io (https://www.gitignore.io) မှ ထိုဘာသာစကားအတွက် ပုံမှန် ignore pattern များကို ရယူပေးမည်ဖြစ်သည်။ `git extras` ကို install လုပ်ပြီး `git alias` သို့မဟုတ် `git ignore` ကဲ့သို့သော tool အချို့ကို စမ်းသုံးကြည့်ပါ။

1. Git GUI ပရိုဂရမ်များသည် အချို့သော အခါများတွင် ကောင်းမွန်သော အရင်းအမြစ်တစ်ခု ဖြစ်နိုင်သည်။ git repo တစ်ခုတွင် gitk (https://git-scm.com/docs/gitk) ကို run ကြည့်ပြီး interface ၏ မတူညီသော အပိုင်းများကို လေ့လာကြည့်ပါ။ ထို့နောက် `gitk --all` ကို run ပါ၊ ကွဲပြားခြားနားချက်များမှာ မည်သည်တို့ နည်း။

1. သင် command line application များကို ကျင့်သုံးသွားပါက GUI tool များသည် ရှုပ်ထွေး/မလိုအပ်ဘဲ ကြီးမားနေသကဲ့သို့ ခံစားရနိုင်သည်။ ၎င်းတို့ နှစ်ခုကြား ကောင်းမွန်သော ညှိနှိုင်းမှုတစ်ခုမှာ command li

```
ne မှ မောင်းနှင်နိုင်ပြီး interactive interface ကိုလည်း ပေးစွမ်းနိုင်သော ncurses အခြေပြု tool
များဖြစ်ကြသည်။ Git တွင် <a href="https://github.com/jonas/tig" class="ext-link">ti
g</a> <span class="ext-url-print">(https://github.com/jonas/tig)</span> ရှိသည်၊ ၎
င်းကို install လုပ်ပြီး repo တစ်ခုတွင် run ကြည့်ပါ။ အသုံးပြုပုံ ဥပမာအချို့ကို <a href="https://
www.atlassian.com/blog/git/git-tig" class="ext-link">ဤနေရာတွင်</a> <span class
="ext-url-print">(https://www.atlassian.com/blog/git/git-tig)</span> တွေ့ရှိနိုင်ပါသ
ည်။




```{=html}
<div class="lecture-links-appendix">
  <h4 class="links-appendix-title">🔗 External Links & References (ပြင်ပ လင့်ခ်များ
နှင့် ကိုးကားချက်များ)</h4>
  <table class="links-table">
    <thead>
      <tr>
        <th style="width: 30%;">Resource / Description</th>
        <th style="width: 56%;">URL</th>
        <th style="width: 14%; text-align: center;">Micro QR</th>
      </tr>
    </thead>
    <tbody>
      <tr>
  <td class="col-label"><strong>1. မြောက်မြားစွာ</strong></td>
  <td class="col-url"><a href="https://en.wikipedia.org/wiki/Comparison_of_vers
ion-control_software" class="appendix-link">https://en.wikipedia.org/wiki/Compa
rison_of_version-control_software</a></td>
  <td class="col-qr"><img src="file:///home/runner/work/missing-semester-my.git
hub.io/missing-semester-my.github.io/ebook/assets/qrcodes/link_dba5de8e_micro.p
ng" alt="QR" class="micro-qr" /></td>
</tr><tr>
  <td class="col-label"><strong>2. git</strong></td>
  <td class="col-url"><a href="https://git-scm.com/" class="appendix-link">http
s://git-scm.com/</a></td>
  <td class="col-qr"><img src="file:///home/runner/work/missing-semester-my.git
hub.io/missing-semester-my.github.io/ebook/assets/qrcodes/link_fb811237_micro.p
ng" alt="QR" class="micro-qr" /></td>
</tr><tr>
  <td class="col-label"><strong>3. Mercurial</strong></td>
  <td class="col-url"><a href="https://www.mercurial-scm.org/" class="appendix-
link">https://www.mercurial-scm.org/</a></td>
  <td class="col-qr"><img src="file:///home/runner/work/missing-semester-my.git
hub.io/missing-semester-my.github.io/ebook/assets/qrcodes/link_49114b40_micro.p
ng" alt="QR" class="micro-qr" /></td>
```
```

```
</tr><tr>
  <td class="col-label"><strong>4. ဤနေရာတွင်</strong></td>
  <td class="col-url"><a href="https://git-scm.com/book/en/v2/Git-Branching-Reb
asing" class="appendix-link">https://git-scm.com/book/en/v2/Git-Branching-Rebas
ing</a></td>
  <td class="col-qr"><img src="file:///home/runner/work/missing-semester-my.git
hub.io/missing-semester-my.github.io/ebook/assets/qrcodes/link_409fb012_micro.p
ng" alt="QR" class="micro-qr" /></td>
</tr><tr>
  <td class="col-label"><strong>5. ဤဆောင်းပါး</strong></td>
  <td class="col-url"><a href="https://www.atlassian.com/git/tutorials/merging-
vs-rebasing" class="appendix-link">https://www.atlassian.com/git/tutorials/merg
ing-vs-rebasing</a></td>
  <td class="col-qr"><img src="file:///home/runner/work/missing-semester-my.git
hub.io/missing-semester-my.github.io/ebook/assets/qrcodes/link_eed4a794_micro.p
ng" alt="QR" class="micro-qr" /></td>
</tr><tr>
  <td class="col-label"><strong>6. ဤဆောင်းပါး</strong></td>
  <td class="col-url"><a href="https://web.archive.org/web/20210106220723/http
s://derekgourlay.com/blog/git-when-to-merge-vs-when-to-rebase/" class="appendix
-link">https://web.archive.org/web/20210106220723/https://derekgourlay.com/blo
g/git-when-to-merge-vs-when-to-rebase/</a></td>
  <td class="col-qr"><img src="file:///home/runner/work/missing-semester-my.git
hub.io/missing-semester-my.github.io/ebook/assets/qrcodes/link_329d0bdf_micro.p
ng" alt="QR" class="micro-qr" /></td>
</tr><tr>
  <td class="col-label"><strong>7. ဤဆောင်းပါး</strong></td>
  <td class="col-url"><a href="https://stackoverflow.com/questions/804115/when-
do-you-use-git-rebase-instead-of-git-merge" class="appendix-link">https://stack
overflow.com/questions/804115/when-do-you-use-git-rebase-instead-of-git-merge</
a></td>
  <td class="col-qr"><img src="file:///home/runner/work/missing-semester-my.git
hub.io/missing-semester-my.github.io/ebook/assets/qrcodes/link_2e111abf_micro.p
ng" alt="QR" class="micro-qr" /></td>
</tr><tr>
  <td class="col-label"><strong>8. ![git အကြောင်း XKCD</strong></td>
  <td class="col-url"><a href="https://imgs.xkcd.com/comics/git.png" class="app
endix-link">https://imgs.xkcd.com/comics/git.png</a></td>
  <td class="col-qr"><img src="file:///home/runner/work/missing-semester-my.git
hub.io/missing-semester-my.github.io/ebook/assets/qrcodes/link_e8d209f8_micro.p
ng" alt="QR" class="micro-qr" /></td>
</tr><tr>
  <td class="col-label"><strong>9. Learn git branching</strong></td>
  <td class="col-url"><a href="https://learngitbranching.js.org/" class="append
ix-link">https://learngitbranching.js.org/</a></td>
  <td class="col-qr"><img src="file:///home/runner/work/missing-semester-my.git
```

```
hub.io/missing-semester-my.github.io/ebook/assets/qrcodes/link_60f84362_micro.p
ng" alt="QR" class="micro-qr" /></td>
</tr><tr>
  <td class="col-label"><strong>10. How to explain git in simple words</strong>
</td>
  <td class="col-url"><a href="https://smusamashah.github.io/blog/2017/10/14/ex
plain-git-in-simple-words" class="appendix-link">https://smusamashah.github.io/
blog/2017/10/14/explain-git-in-simple-words</a></td>
  <td class="col-qr"><img src="file:///home/runner/work/missing-semester-my.git
hub.io/missing-semester-my.github.io/ebook/assets/qrcodes/link_ac81b1b9_micro.p
ng" alt="QR" class="micro-qr" /></td>
</tr><tr>
  <td class="col-label"><strong>11. Git from the bottom up</strong></td>
  <td class="col-url"><a href="https://jwiegley.github.io/git-from-the-bottom-u
p/" class="appendix-link">https://jwiegley.github.io/git-from-the-bottom-up/</a
></td>
  <td class="col-qr"><img src="file:///home/runner/work/missing-semester-my.git
hub.io/missing-semester-my.github.io/ebook/assets/qrcodes/link_0dcb5949_micro.p
ng" alt="QR" class="micro-qr" /></td>
</tr><tr>
  <td class="col-label"><strong>12. Git for computer scientists</strong></td>
  <td class="col-url"><a href="https://eagain.net/articles/git-for-computer-sci
entists/" class="appendix-link">https://eagain.net/articles/git-for-computer-sc
ientists/</a></td>
  <td class="col-qr"><img src="file:///home/runner/work/missing-semester-my.git
hub.io/missing-semester-my.github.io/ebook/assets/qrcodes/link_634c0c10_micro.p
ng" alt="QR" class="micro-qr" /></td>
</tr><tr>
  <td class="col-label"><strong>13. Oh shit, git!</strong></td>
  <td class="col-url"><a href="https://ohshitgit.com/" class="appendix-link">ht
tps://ohshitgit.com/</a></td>
  <td class="col-qr"><img src="file:///home/runner/work/missing-semester-my.git
hub.io/missing-semester-my.github.io/ebook/assets/qrcodes/link_44507355_micro.p
ng" alt="QR" class="micro-qr" /></td>
</tr><tr>
  <td class="col-label"><strong>14. The Pro Git book</strong></td>
  <td class="col-url"><a href="https://git-scm.com/book/en/v2" class="appendix-
link">https://git-scm.com/book/en/v2</a></td>
  <td class="col-qr"><img src="file:///home/runner/work/missing-semester-my.git
hub.io/missing-semester-my.github.io/ebook/assets/qrcodes/link_fadfb26c_micro.p
ng" alt="QR" class="micro-qr" /></td>
</tr><tr>
  <td class="col-label"><strong>15. ဤဆောင်းပါး</strong></td>
  <td class="col-url"><a href="https://help.github.com/articles/removing-sensit
ive-data-from-a-repository/" class="appendix-link">https://help.github.com/arti
cles/removing-sensitive-data-from-a-repository/</a></td>
```

```
  <td class="col-qr"><img src="file:///home/runner/work/missing-semester-my.git
hub.io/missing-semester-my.github.io/ebook/assets/qrcodes/link_2267a7d9_micro.p
ng" alt="QR" class="micro-qr" /></td>
</tr><tr>
  <td class="col-label"><strong>16. Git-LFS</strong></td>
  <td class="col-url"><a href="https://git-lfs.github.com/" class="appendix-lin
k">https://git-lfs.github.com/</a></td>
  <td class="col-qr"><img src="file:///home/runner/work/missing-semester-my.git
hub.io/missing-semester-my.github.io/ebook/assets/qrcodes/link_5db28766_micro.p
ng" alt="QR" class="micro-qr" /></td>
</tr><tr>
  <td class="col-label"><strong>17. git extras</strong></td>
  <td class="col-url"><a href="https://github.com/tj/git-extras/blob/master/Com
mands.md" class="appendix-link">https://github.com/tj/git-extras/blob/master/Co
mmands.md</a></td>
  <td class="col-qr"><img src="file:///home/runner/work/missing-semester-my.git
hub.io/missing-semester-my.github.io/ebook/assets/qrcodes/link_eaee3ce1_micro.p
ng" alt="QR" class="micro-qr" /></td>
</tr><tr>
  <td class="col-label"><strong>18. gitignore.io</strong></td>
  <td class="col-url"><a href="https://www.gitignore.io" class="appendix-link">
https://www.gitignore.io</a></td>
  <td class="col-qr"><img src="file:///home/runner/work/missing-semester-my.git
hub.io/missing-semester-my.github.io/ebook/assets/qrcodes/link_4b75a8ac_micro.p
ng" alt="QR" class="micro-qr" /></td>
</tr><tr>
  <td class="col-label"><strong>19. gitk</strong></td>
  <td class="col-url"><a href="https://git-scm.com/docs/gitk" class="appendix-l
ink">https://git-scm.com/docs/gitk</a></td>
  <td class="col-qr"><img src="file:///home/runner/work/missing-semester-my.git
hub.io/missing-semester-my.github.io/ebook/assets/qrcodes/link_765f1834_micro.p
ng" alt="QR" class="micro-qr" /></td>
</tr><tr>
  <td class="col-label"><strong>20. tig</strong></td>
  <td class="col-url"><a href="https://github.com/jonas/tig" class="appendix-li
nk">https://github.com/jonas/tig</a></td>
  <td class="col-qr"><img src="file:///home/runner/work/missing-semester-my.git
hub.io/missing-semester-my.github.io/ebook/assets/qrcodes/link_ae930c0e_micro.p
ng" alt="QR" class="micro-qr" /></td>
</tr><tr>
  <td class="col-label"><strong>21. ဤနေရာတွင်</strong></td>
  <td class="col-url"><a href="https://www.atlassian.com/blog/git/git-tig" clas
s="appendix-link">https://www.atlassian.com/blog/git/git-tig</a></td>
  <td class="col-qr"><img src="file:///home/runner/work/missing-semester-my.git
hub.io/missing-semester-my.github.io/ebook/assets/qrcodes/link_12c887a0_micro.p
ng" alt="QR" class="micro-qr" /></td>
```

```
  </tr>
      </tbody>
    </table>
  </div>
```

Virtual Machines နှင့် Containers များ

▶ LECTURE VIDEO REFERENCE

Virtual Machines နှင့် Containers များ

Scan the QR code below using your mobile camera or click the link to watch this full lecture online:

Direct Online Link:

https://www.youtube.com/watch?v=LJ9ki5zq6Ik

Virtual Machines

Virtual machine များသည် အတုပြုလုပ်ထားသော ကွန်ပျူတာများ (simulated computers) ဖြစ်ကြသည်။ သင်သည် guest virtual machine တစ်ခုတွင် operating system အချို့နှင့် configuration များကို ပြင်ဆင်သတ်မှတ်ကာ မိမိ၏ host ပတ်ဝန်းကျင်ကို ထိခိုက်မှုမရှိဘဲ အသုံးပြုနိုင်ပါသည်။

ဤသင်တန်းအတွက် သင်သည် ဘေးအန္တရာယ် ကင်းရှင်းစွာဖြင့် operating system များ၊ ဆော့ဖ်ဝဲလ်များနှင့် configuration များကို စမ်းသပ်ရန် VM များကို အသုံးပြုနိုင်ပါသည်—ယင်းက သင်၏ အဓိက development ပတ်ဝန်းကျင်ကို ထိခိုက်စေမည် မဟုတ်ပါ။

ယေဘုယျအားဖြင့် VM များတွင် အသုံးပြုပုံ အများအပြား ရှိပါသည်။ ၎င်းတို့ကို သီးခြား operating system တစ်ခုပေါ်တွင်သာ run နိုင်သော ဆော့ဖ်ဝဲလ်များကို run ရန်အတွက် အလွန် အသုံးများကြပါသည် (ဥပမာ- Windows-သီးသန့် ဆော့ဖ်ဝဲလ်များကို run ရန် Linux ပေါ်တွင် Windows VM တစ်ခုကို အသုံးပြုခြင်း)။ ထို့အပြင် အန္တရာယ်ရှိနိုင်သော (malicious) ဆော့ဖ်ဝဲလ်များကို စမ်းသပ်ရန်အတွက်လည်း ၎င်းတို့ကို မကြာခဏ အသုံးပြုကြပါသည်။

အသုံးဝင်သော အင်္ဂါရုပ်များ

- **Isolation (သီးခြားခွဲထုတ်ထားခြင်း)**: hypervisor များသည် guest ကို host မှ သီးခြားခွဲထုတ်ရာတွင် အတော်ပင် ကောင်းမွန်စွာ လုပ်ဆောင်ပေးနိုင်သောကြောင့် ဘင်္ဂပါနိုင်သော သို့မဟုတ် မယုံကြည်ရသော ဆော့ဖ်ဝဲလ်များကို VM များ အသုံးပြု၍ စိတ်ချလက်ချ run နိုင်ပါသည်။
- **Snapshots (ဓာတ်ပုံရိုက်ကူးသကဲ့သို့ သိမ်းဆည်းခြင်း)**: သင်၏ virtual machine ၏ “snapshots” များကို ရယူထားနိုင်ပြီး စက်တစ်ခုလုံး၏ အခြေအနေ (disk, memory စသည်ဖြင့်) ကို သိမ်းဆည်းကာ စက်တွင် ပြောင်းလဲမှုများ ပြုလုပ်ပြီးနောက် ယခင်အခြေအနေသို့ ပြန်လည် ထိန်းသိမ်း (restore ပြုလုပ်) နိုင်ပါသည်။ ၎င်းသည် အခြားအရာများအပြင် ပျက်စီးစေနိုင်သော လုပ်ဆောင်ချက်များကို စမ်းသပ်ကြည့်ရှုရန်အတွက် အလွန် အသုံးဝင်ပါသည်။

အားနည်းချက်များ

Virtual machine များသည် အမှန်တကယ် စက်ပစ္စည်း (bare metal) ပေါ်တွင် တိုက်ရိုက် run ခြင်းထက် ယေဘုယျအားဖြင့် ပိုမို နှေးကွေးသောကြောင့် အချို့သော application များအတွက် အဆင်မပြေနိုင်ပါ။

ပြင်ဆင်သတ်မှတ်ခြင်း (Setup)

- **Resources**: host စက်နှင့် မျှဝေသုံးစွဲရပါသည်၊ ရုပ်ပိုင်းဆိုင်ရာ resource များကို ခွဲဝေပေးသည့်အခါ ဤအချက်ကို သတိပြုပါ။
- **Networking**: ရွေးချယ်စရာ နည်းလမ်းများစွာ ရှိပြီး၊ မူလပါဝင်သော NAT သည် အသုံးပြုမှု အများစုအတွက် ကောင်းမွန်စွာ အလုပ်လုပ်သင့်ပါသည်။
- **Guest addons**: hypervisor အများအပြားသည် host စနစ်နှင့် ပိုမို အဆင်ပြေစွာ ပေါင်းစပ်အသုံးပြုနိုင်ရန် guest အတွင်း၌ ဆော့ဖ်ဝဲလ် တပ်ဆင်ပေးနိုင်ပါသည်။ ဖြစ်နိုင်ပါက အဆိုပါ addon များကို အသုံးပြုသင့်ပါသည်။

အရင်းအမြစ်များ (Resources)

- Hypervisor များ

```
- <a href="https://www.virtualbox.org/" class="ext-link">VirtualBox</a> <span c
lass="ext-url-print">(https://www.virtualbox.org/)</span> (open-source)
- <a href="https://virt-manager.org/" class="ext-link">Virt-manager</a> <span c
lass="ext-url-print">(https://virt-manager.org/)</span> (open-source, KVM virtu
al machine များနှင့် LXC container များကို စီမံခန့်ခွဲပေးပါသည်)
- <a href="https://www.vmware.com/" class="ext-link">VMWare</a> <span class="ex
t-url-print">(https://www.vmware.com/)</span> (commercial, IS&T မှတစ်ဆင့် <a href
="https://ist.mit.edu/vmware-fusion" class="ext-link">MIT ကျောင်းသားများအတွက်</a>
<span class="ext-url-print">(https://ist.mit.edu/vmware-fusion)</span> ရယူနိုင်ပါသ
ည်)
```

အကယ်၍ သင်သည် လူသိများသော hypervisor/VM များနှင့် ရင်းနှီးပြီးသားဖြစ်ပါက command line ဖြင့် ပိုမို လွယ်ကူစွာ မည်သို့ ပြုလုပ်နိုင်သည်ကို ပိုမို လေ့လာလိုပေလိမ့်မည်။ ရွေးချယ်စရာ တစ်ခုမှာ ကွဲပြားသော virtualization provider/hypervisor အများအပြားကို စီမံခန့်ခွဲနိုင်စေသည့် **libvirt** (https://wiki.libvirt.org/page/UbuntuKVMWalkthrough) toolkit ဖြစ်ပါသည်။

လေ့ကျင့်ခန်းများ

1. Hypervisor တစ်ခုကို ဒေါင်းလုဒ်လုပ်ပြီး install လုပ်ပါ။
2. Virtual machine အသစ်တစ်ခု ဖန်တီးပြီး Linux distribution တစ်ခု (ဥပမာ- **Debian** (https://www.debian.org/)) ကို install လုပ်ပါ။

3. Snapshots များကို စမ်းသပ်ကြည့်ပါ။ `sudo rm -rf --no-preserve-root /` ကို run ခြင်းကဲ့သို့ သင် အမြဲ စမ်းကြည့်ချင်ခဲ့သည့် အရာများကို စမ်းသပ်ကြည့်ပြီး အလွယ်တကူ ပြန်လည်ရယူနိုင်ခြင်း ရှိမရှိ ကြည့်ပါ။

4. **fork-bomb** (https://en.wikipedia.org/wiki/Fork_bomb) (`:(){ :|:& };:`) ဆိုသည်မှာ အဘယ်နည်းကို ဖတ်ရှုပြီး resource isolation (CPU, Memory, စသည်) အလုပ်လုပ်ကြောင်း တွေ့ရှိနိုင်ရန် ၎င်းကို VM ပေါ်တွင် run ကြည့်ပါ။

5. Guest addons များကို install လုပ်ပြီး မတူညီသော windowing mode များ၊ file sharing နှင့် အခြား အင်္ဂါရုပ်များကို စမ်းသပ်ကြည့်ပါ။

Containers

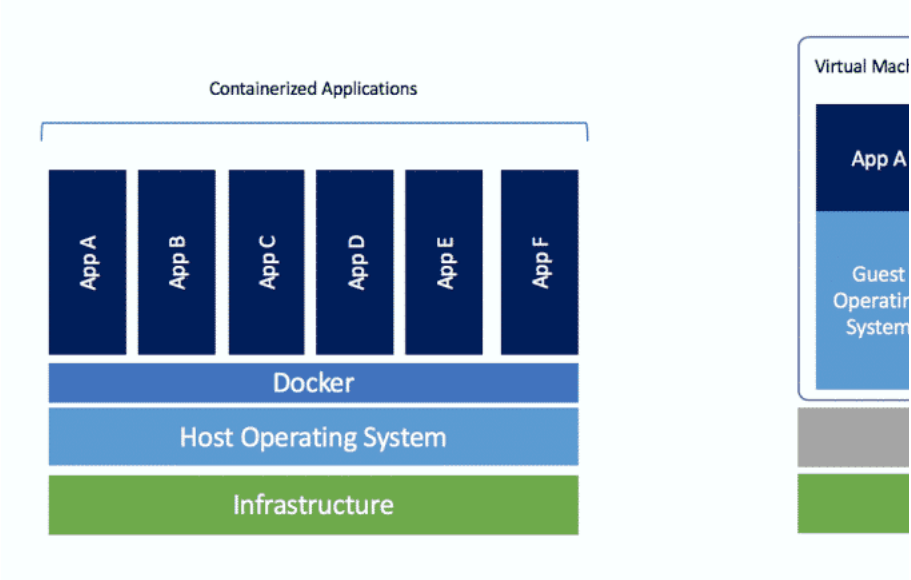
Virtual Machine များသည် အတော်အတန် လေးလံပါသည် (heavy-weight)၊ အကယ်၍ သင်သည် စက်များကို အလိုအလျောက် စနစ်ဖြင့် ဖန်တီးလိုပါက မည်သို့ ပြုလုပ်မည်နည်း။ Containers များ ရောက်ရှိလာပါပြီ။

- Amazon Firecracker
- Docker
- rkt
- lxc

Containers ဆိုသည်မှာ virtual file system, virtual network interfaces, chroots, virtual memory နည်းလမ်းများ စသည့် အမျိုးမျိုးသော Linux လုံခြုံရေး အင်္ဂါရုပ်များကို အတူတကွ ပေါင်းစပ်ထားခြင်း *အများစု* သာ ဖြစ်ပြီး၊ ၎င်းတို့ အားလုံး ပေါင်းစပ်၍ virtualization ပုံစံမျိုး ထွက်ပေါ်လာခြင်း ဖြစ်ပါသည်။

VM တစ်ခုကဲ့သို့ လုံခြုံမှု သို့မဟုတ် သီးခြားခွဲထုတ်ထားမှု မရှိသော်လည်း အလွန် နီးစပ်ပြီး ပိုမို ကောင်းမွန်လာနေပါသည်။ ယေဘုယျအားဖြင့် စွမ်းဆောင်ရည် ပိုမို မြင့်မားပြီး စတင်ရန် အလွန် မြန်ဆန်သော်လည်း အမြဲတမ်းတော့ မဟုတ်ပါ။

Operating system တစ်ခုလုံး၏ ကော်ပီကို run သည့် VM များနှင့် မတူဘဲ container များသည် host ၏ linux kernel ကို မျှဝေသုံးစွဲသည့် အချက်ကြောင့် စွမ်းဆောင်ရည် ပိုမို မြင့်မားလာခြင်း ဖြစ်ပါသည်။ သို့သော် Windows/macOS ပေါ်တွင် linux container များကို run နေပါက ယင်းနှစ်ခုကြားတွင် ကြားခံအလွှာ (middle layer) အဖြစ် Linux VM တစ်ခု ဖွင့်ထားရန် လိုအပ်မည်ဖြစ်ကြောင်း သတိပြုပါ။



Docker containers နှင့် Virtual Machines များကြား နှိုင်းယှဉ်ချက်။ Credit: blog.docker.com

စံသတ်မှတ်ထားသော စီမံပြင်ဆင်မှုတစ်ခုတွင် အလိုအလျောက် အလုပ်တစ်ခုကို run လိုသည့်အခါ Container များသည် အသုံးဝင်ပါသည်။

- Build systems
- Development environments
- Pre-packaged servers
- မယုံကြည်ရသော ပရိုဂရမ်များကို run ခြင်း
  - ကျောင်းသားများ၏ တင်ပြချက်များကို အမှတ်ပေးခြင်း
  - (အချို့သော) cloud computing
- Continuous integration
  - Travis CI
  - GitHub Actions

ထို့အပြင် Docker ကဲ့သို့သော container ဆော့ဖ်ဝဲလ်များကို **dependency hell** (https://en.wikipedia.org/wiki/Dependency_hell) အတွက် ဖြေရှင်းချက်အဖြစ်လည်း ကျယ်ကျယ်ပြန့်ပြန့် အသုံးပြုကြ

ပါသည်။ အကယ်၍ စက်တစ်ခုသည် ပဋိပက္ခဖြစ်နေသော dependency များရှိသည့် service အမြောက်အများကို run ရန် လိုအပ်ပါက container များကို အသုံးပြု၍ ၎င်းတို့ကို သီးခြား ခွဲထုတ်ထားနိုင်ပါသည်။

ပုံမှန်အားဖြင့် သင်သည် သင်၏ container ကို မည်သို့ တည်ဆောက်ရမည်ကို သတ်မှတ်သည့် ဖိုင်တစ်ခုကို ရေးသားရပါသည်။ သင်သည် အနည်းဆုံး ပါဝင်သော *base image* တစ်ခု (Alpine Linux ကဲ့သို့) ဖြင့် စတင်ပြီး၊ ယင်းနောက် သင်လိုချင်သည့် ပတ်ဝန်းကျင်ကို ပြင်ဆင်သတ်မှတ်ရန် run ရမည့် command စာရင်းများ (package များ install လုပ်ခြင်း၊ ဖိုင်များ ကူးယူခြင်း၊ build လုပ်ခြင်း၊ config ဖိုင်များ ရေးသားခြင်း စသည်) ကို ရေးသားရပါသည်။ ပုံမှန်အားဖြင့် ရရှိနိုင်မည့် အပြင်ဘက် port များကို သတ်မှတ်ပေးသည့် နည်းလမ်းလည်း ပါဝင်ပြီး container စတင်သည့်အခါ မည်သည့် command ကို run ရမည်ကို ညွှန်ကြားသည့် *entrypoint* တစ်ခု (အမှတ်ပေး script ကဲ့သို့) လည်း ပါဝင်ပါသည်။

Code repository ဝဘ်ဆိုက်များ (**GitHub** (https://github.com/) ကဲ့သို့) နှင့် ဆင်တူစွာပင်၊ ဆော့ဖ်ဝဲလ် service အများအပြားအတွက် အလွယ်တကူ deploy လုပ်နိုင်သော prebuilt image များ ရှိသည့် container repository ဝဘ်ဆိုက်များ (**DockerHub** (https://hub.docker.com/) ကဲ့သို့) လည်း ရှိပါသည်။

လေ့ကျင့်ခန်းများ

1. Container ဆော့ဖ်ဝဲလ် တစ်ခု (Docker, LXC, ...) ကို ရွေးချယ်ပြီး ရိုးရှင်းသော Linux image တစ်ခုကို install လုပ်ပါ။ ၎င်းအတွင်းသို့ SSH ဝင်ကြည့်ရန် စမ်းသပ်ပါ။

2. လူကြိုက်များသော ဝဘ်ဆာဗာ (nginx, apache, ...) အတွက် prebuilt container image တစ်ခုကို ရှာဖွေပြီး ဒေါင်းလုဒ်လုပ်ပါ။

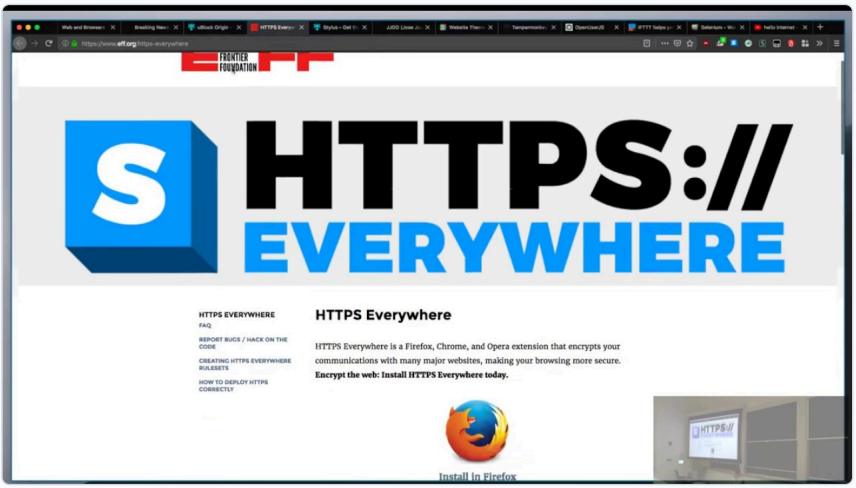
🔗 External Links & References (ပြင်ပ လင့်ခ်များနှင့် ကိုးကားချက်များ)

Resource / Description	URL	Micro QR
1. VirtualBox	https://www.virtualbox.org/	
2. Virt-manager	https://virt-manager.org/	
3. VMWare	https://www.vmware.com/	
4. MIT ကျောင်းသားများအတွက်	https://ist.mit.edu/vmware-fusion	
5. libvirt	https://wiki.libvirt.org/page/UbuntuKVMWalkthrough	
6. Debian	https://www.debian.org/	
7. fork-bomb	https://en.wikipedia.org/wiki/Fork_bomb	
8. dependency hell	https://en.wikipedia.org/wiki/Dependency_hell	
9. GitHub	https://github.com/	
10. DockerHub	https://hub.docker.com/	

Web နှင့် Web Browser များ

▶ LECTURE VIDEO REFERENCE

Web နှင့် Web Browser များ



Scan the QR code below using your mobile camera or click the link to watch this full lecture online:

Direct Online Link:

https://www.youtube.com/watch?v=XpZO3S8odec

Terminal ပြီးရင် Web Browser ဟာ သင်အချိန်တော်တော်များများ အသုံးပြုရမယ့် Tool တစ်ခုဖြစ်ပါတယ်။ ဒါကြောင့်မို့လို့ သူ့ကို ထိရောက်ပြီး အကျိုးရှိရှိ ဘယ်လိုအသုံးပြုရမလဲဆိုတာ လေ့လာသင်ယူထားတာ ထိုက်တန်ပါတယ်။

Shortcuts (ဖြတ်လမ်းနည်းများ)

မိမိရဲ့ Browser ထဲမှာ ကလစ်လိုက်နှိပ်နေခြင်းဟာ အမြဲတမ်း အမြန်ဆုံးနည်းလမ်း မဟုတ်ပါဘူး။ အသုံးများတဲ့ Shortcut တွေကို ယှဉ်တွဲသိရှိထားခြင်းဟာ ရေရှည်မှာ တကယ်ပဲ အကျိုးရှိစေမှာ ဖြစ်ပါတယ်။

- Link တစ်ခုပေါ်မှာ `Middle Button Click` (Mouse ဘီးနှိပ်ခြင်း) နှိပ်ပါက Tab အသစ်တစ်ခုမှာ ဖွင့်ပေးပါမည်။
- `Ctrl+T` - Tab အသစ်တစ်ခု ဖွင့်ရန်။
- `Ctrl+Shift+T` - မကြာသေးမီက ပိတ်လိုက်သော Tab ကို ပြန်ဖွင့်ရန်။
- `Ctrl+L` - Search bar ထဲမှ စာသားများကို ရွေးချယ်ရန်။
- `Ctrl+F` - Webpage အတွင်း ရှာဖွေရန်။ အကယ်၍ သင်သည် ဤသို့မကြာခဏ ရှာဖွေလေ့ရှိပါက Search များတွင် Regular Expression များကို ပံ့ပိုးပေးသည့် Extension တစ်ခုကို အသုံးပြုခြင်းဖြင့် အကျိုးရှိနိုင်ပါတယ်။

Search operators (ရှာဖွေမှု သင်္ကေတများ)

Google သို့မဟုတ် DuckDuckGo ကဲ့သို့သော Web Search Engine များသည် ပိုမိုတိကျ အသေးစိတ်ကျသော ရှာဖွေမှုများ ပြုလုပ်နိုင်ရန် Search Operator များကို ပံ့ပိုးပေးထားပါတယ်-

- `"bar foo"` - bar foo ဟူသော စကားစု အတိအကျ ပါဝင်မှုကို သတ်မှတ်ရှာဖွေပေးသည်။
- `foo site:bar.com` - bar.com ဝက်ဘ်ဆိုက် အတွင်း၌သာ foo ကို ရှာဖွေပေးသည်။
- `foo -bar` - bar ဟူသော စကားလုံး ပါဝင်သည့် ရလဒ်များကို ရှာဖွေမှုမှ ဖယ်ထုတ်ပေးသည်။
- `foobar filetype:pdf` - ထို File Extension ရှိသည့် ဖိုင်များကို ရှာဖွေပေးသည်။
- `(foo|bar)` - foo သို့မဟုတ် bar ပါဝင်သည့် ကိုက်ညီမှုများကို ရှာဖွေပေးသည်။

Google (https://ahrefs.com/blog/google-advanced-search-operators/) နှင့် DuckDuckGo (https://duck.co/help/results/syntax) ကဲ့သို့သော လူကြိုက်များသည့် Engine များအတွက် ပိုမိုပြည့်စုံသည့် စာရင်းများကို သက်ဆိုင်ရာ Link များတွင် ဝင်ရောက်ကြည့်ရှုနိုင်ပါတယ်။

Searchbar (ရှာဖွေမှုဘား)

Searchbar ဟာလည်း စွမ်းဆောင်ရည်ထက်မြက်တဲ့ Tool တစ်ခု ဖြစ်ပါတယ်။ Browser အများစုဟာ ဝက်ဘ်ဆိုက်များမှ Search Engine များကို အလိုအလျောက် ရယူမှတ်သားထားနိုင်ပါတယ်။ Keyword argument ကို ပြင်ဆင်ခြင်းဖြင့်-

- Google Chrome တွင် chrome://settings/searchEngines ၌ ရှိသည်။
- Firefox တွင် about:preferences#search ၌ ရှိသည်။

ဥပမာအားဖြင့် `y SOME SEARCH TERMS` ဟု ရိုက်ထည့်ရုံဖြင့် YouTube ထဲတွင် တိုက်ရိုက် ရှာဖွေနိုင်အောင် ပြုလုပ်နိုင်ပါတယ်။

ထို့ပြင် သင့်ထံတွင် ကိုယ်ပိုင် Domain ရှိပါက Registrar မှတစ်ဆင့် Subdomain Forward ပြုလုပ်ထားနိုင်ပါတယ်။ ဥပမာအားဖြင့် ကျွန်တော်သည် `https://ht.josejg.com` ကို ဤသင်တန်း ဝက်ဘ်ဆိုက်သို့ ချိန်ညှိ (map) ထားပါတယ်။ ဤနည်းဖြင့် `ht.` ဟု ရိုက်ထည့်လိုက်သည်နှင့် Searchbar က အလိုအလျောက် ဖြည့်စွက် (autocomplete) ပေးသွားမှာ ဖြစ်ပါတယ်။ ဤစနစ်၏ အခြားကောင်းမွန်သည့် အချက်တစ်ခုမှာ Bookmark များနှင့် မတူဘဲ Browser တိုင်းတွင် အလုပ်လုပ်မည် ဖြစ်ခြင်းပင် ဖြစ်ပါတယ်။

Privacy extensions (သီးသန့်လုံခြုံရေးဆိုင်ရာ Extension များ)

ယနေ့ခေတ်တွင် ကြော်ငြာများကြောင့် Web ကြည့်ရှုရသည်မှာ စိတ်ရှုပ်စရာ ကောင်းလာပြီး Tracker များကြောင့် ကိုယ်ရေးအချက်အလက် ကျူးလွန်ခံရမှုများ ရှိလာပါတယ်။ ထို့ပြင် ကောင်းမွန်သည့် Adblocker တစ်ခုသည် ကြော်ငြာအများစုကို ပိတ်ပင်ပေးရုံသာမက သံသယဖြစ်ဖွယ်နှင့် မကောင်းသော မဲလ်ဝဲ ဝက်ဘ်ဆိုက်များကိုလည်း အများသုံး Blacklist စာရင်းများတွင် ပါဝင်သောကြောင့် ပိတ်ပင်ပေးသွားမည် ဖြစ်ပါတယ်။ ၎င်းတို့သည် ပြုလုပ်ရသည့် Request အရေအတွက်ကို လျှော့ချပေးခြင်းဖြင့် ဝက်ဘ်ဆိုက် ပွင့်သည့် ကြာချိန် (page load time) ကိုလည်း အချို့နေရာများတွင် ပိုမိုမြန်ဆန်စေပါတယ်။ အကြံပြုလိုသည့် Extension အချို့မှာ-

- **uBlock origin** (**Chrome** (https://chrome.google.com/webstore/detail/ublock-origin/cjpalhdlnbpafiamejdnhcphjbkeiagm), **Firefox** (https://addons.mozilla.org/en-US/firefox/addon/ublock-origin/)) - သတ်မှတ်ထားသော စည်းမျဉ်းများ (rules) ပေါ်မူတည်၍ ကြော်ငြာများနှင့် Tracker များကို ပိတ်ပင်ပေးသည်။ သင်၏ ဒေသ သို့မဟုတ် သုံးစွဲမှု အလေ့အထများပေါ်မူတည်၍ ထပ်မံဖွင့်လှစ်နိုင်သောကြောင့် Settings ထဲရှိ ဖွင့်ထားသော Blacklist စာရင်းများကို ကြည့်ရှုစစ်ဆေးသင့်ပါတယ်။ **အင်တာနက်ပေါ်ရှိ** (https://github.com/gorhill/uBlock/wiki/Filter-lists-from-around-the-web) Filter စာရင်းများကိုလည်း ထည့်သွင်းအသုံးပြုနိုင်ပါတယ်။
- **Privacy Badger** (https://privacybadger.org/) - Tracker များကို အလိုအလျောက် ထောက်လှမ်း၍ ပိတ်ပင်ပေးသည်။ ဥပမာ သင်သည် ဝက်ဘ်ဆိုက်တစ်ခုမှ တစ်ခုသို့ ကူးပြောင်း ကြည့်ရှုသည့်အခါ ကြော်ငြာကုမ္ပဏီများသည် သင်ဝင်ရောက်သည့် ဆိုက်များကို စောင့်ကြည့်မှတ်သားပြီး သင့်ဆိုင်ရာ Profile တစ်ခု တည်ဆောက်ကြသည်။

- **HTTPS everywhere** (https://www.eff.org/https-everywhere) - ရရှိနိုင်ပါက ဝက်ဘ်ဆိုက်၏ HTTPS Version သို့ အလိုအလျောက် ပြန်လည်ညွှန်းပို့ (redirect) ပေးသည့် ကောင်းမွန်သော Extension တစ်ခု ဖြစ်သည်။

ဤကဲ့သို့သော Addon များအကြောင်း ပိုမို၍ ဤနေရာတွင် (https://www.privacytools.io/privacy-browser-addons/) လေ့လာနိုင်ပါတယ်။

Style customization (ဒီဇိုင်း ပုံစံ ပြင်ဆင်ခြင်း)

Web Browser များသည် သင်၏ *ကွန်ပျူတာစက်* ပေါ်တွင် အလုပ်လုပ်နေသည့် အခြားသော Software တစ်ခုမျှသာ ဖြစ်သောကြောင့် ၎င်းတို့ ဘာပြသရမည်၊ ဘယ်လို ပြုမူဆောင်ရွက်ရမည် ဆိုသည်ကို နောက်ဆုံး ဆုံးဖြတ်ခွင့်မှာ သင့်တွင် ရှိပါတယ်။ ထိုသို့ ပြုလုပ်နိုင်သည့် ဥပမာတစ်ခုမှာ Custom Style များ ပြုလုပ်ခြင်း ဖြစ်ပါတယ်။ Browser များသည် Webpage တစ်ခု၏ ဒီဇိုင်းပုံစံကို ဖော်ပြရန် (render) Cascading Style Sheets (အတိုကောက် CSS) ကို အသုံးပြုကြပါတယ်။

ဝက်ဘ်ဆိုက်တစ်ခု၏ Source Code ကို Inspect ပြုလုပ်ခြင်းဖြင့် ၎င်း၏ အကြောင်းအရာများနှင့် ဒီဇိုင်းပုံစံများကို ယာယီ ပြောင်းလဲကြည့်ရှုနိုင်ပါတယ်။ (ဒါကြောင့်လည်း Webpage ၏ Screenshot များကို မည်သည့် အခါမျှ ရာနှုန်းပြည့် မယုံကြည်သင့်သည့် အကြောင်းရင်း ဖြစ်ပါတယ်)။

အကယ်၍ သင်သည် Webpage တစ်ခု၏ Style Settings များကို အမြဲတမ်း အစားထိုး (override) ပြုလုပ်ရန် သင်၏ Browser အား ခိုင်းစေလိုပါက Extension တစ်ခုကို အသုံးပြုရန် လိုအပ်မည် ဖြစ်သည်။ ကျွန်ုပ်တို့ အကြံပြုလိုသည်မှာ **Stylus** (https://github.com/openstyles/stylus) (**Firefox** (https://addons.mozilla.org/en-US/firefox/addon/styl-us/), **Chrome** (https://chrome.google.com/webstore/detail/stylus/clngdbkpkpeebahjckkjfobafhncgmne?hl=en)) ဖြစ်ပါတယ်။

ဥပမာအားဖြင့် သင်တန်း ဝက်ဘ်ဆိုက်အတွက် အောက်ပါ Style ကို ရေးသားနိုင်ပါတယ်-

```
body {
    background-color: #2d2d2d;
    color: #eee;
    font-family: Fira Code;
    font-size: 16pt;
}

a:link {
    text-decoration: none;
    color: #0a0;
}
```

ထို့ပြင် Stylus သည် အခြားသူများ ရေးသားထားပြီး **userstyles.org** (https://userstyles.org/) တွင် တင်ထားသော Style များကိုလည်း ရှာဖွေပေးနိုင်ပါတယ်။ လူသိများသော ဝက်ဘ်ဆိုက် အများစုတွင် Dark Theme အပြင်အဆင် အနည်းဆုံး တစ်ခု သို့မဟုတ် တစ်ခုထက်မက ရှိတတ်ကြပါတယ်။ FYI အနေဖြင့် Stylish ကို အသုံးမပြုသင့်ပါ။ အဘယ်ကြောင့်ဆိုသော် အသုံးပြုသူ၏ ဒေတာများကို ပေါက်ကြားစေကြောင်း တွေ့ရှိထားရသောကြောင့် ဖြစ်ပြီး အသေးစိတ်ကို ဤနေရာတွင် (https://arstechnica.com/information-technology/2018/07/stylish-extension-with-2m-downloads-banished-for-tracking-every-site-visit/) ဖတ်ရှုနိုင်ပါတယ်။

Functionality Customization (လုပ်ဆောင်ချက်များ ပုံစံပြင်ဆင်ခြင်း)

ဒီဇိုင်းပုံစံကို ပြင်ဆင်နိုင်သကဲ့သို့ပင် ဝက်ဘ်ဆိုက်တစ်ခု၏ လုပ်ဆောင်ပုံ ပြုမူပုံကိုလည်း Custom JavaScript ရေးသားပြီး **Tampermonkey** (https://tampermonkey.net/) ကဲ့သို့သော Browser Extension ဖြင့် ချိတ်ဆက် အသုံးပြုကာ ပြောင်းလဲနိုင်ပါတယ်။

ဥပမာ အောက်ပါ Script သည် J နှင့် K Key များကို အသုံးပြု၍ Vim ကဲ့သို့ သွားလာလှုပ်ရှားမှု (navigation) ပြုလုပ်နိုင်စေပါတယ်-

```
// ==UserScript==
// @name         VIM HT
// @namespace    http://tampermonkey.net/
// @version      0.1
// @description  Vim JK for our website
// @author       You
// @match        https://hacker-tools.github.io/*
// @grant        none
// ==UserScript==


(function() {
    'use strict';

    window.onkeyup = function(e) {
        var key = e.keyCode ? e.keyCode : e.which;

        if (key == 74) { // J is key 74
            window.scrollBy(0,500);;
        }else if (key == 75) { // K is key 75
            window.scrollBy(0,-500);;
        }
    }
})();
```

[OpenUserJS](https://openuserjs.org/) နှင့် [Greasy Fork](https://greasyfork.org/en) ကဲ့သို့သော Script Repository များလည်း ရှိကြပါတယ်။ သို့သော်လည်း သတိပြုရမည်မှာ အခြားသူများ၏ User Script များကို ထည့်သွင်းခြင်းသည် သင့်ခရက်ဒစ်ကတ် နံပါတ်များကို ခိုးယူခြင်းကဲ့သို့သော မည်သည့်အရာကိုမဆို ပြုလုပ်နိုင်သဖြင့် အလွန် အန္တရာယ်များနိုင်ပါတယ်။ သင့်ကိုယ်တိုင် စာကြောင်း တစ်ကြောင်းချင်းစီ ဖတ်ကြည့်ပြီး ၎င်း ပြုလုပ်သည်များကို နားလည်ကာ သံသယဖြစ်ဖွယ် မရှိကြောင်း သေချာစွာ မသိမချင်း မည်သည့် Script ကိုမျှ မထည့်သွင်းပါနှင့်။ သင် မဖတ်နိုင်သော Minified သို့မဟုတ် Obfuscated (ရှုပ်ထွေးအောင် ဖုံးကွယ်ထားသော) Code များ ပါဝင်သည့် Script ကို မည်သည့်အခါမျှ မထည့်သွင်းပါနှင့်။

Web APIs

Web service များတွင် Web Request များ ပြုလုပ်ပြီး သက်ဆိုင်ရာ ဝန်ဆောင်မှုများနှင့် ချိတ်ဆက်လုပ်ဆောင်နိုင်ရန် Web API ဟုခေါ်သော Application Interface များကို ပံ့ပိုးပေးလာကြခြင်းမှာ ပိုမို ခေတ်စားလာခဲ့ပါတယ်။ ဤခေါင်းစဉ်နှင့် ပတ်သက်ပြီး ပိုမို အသေးစိတ်ကျသော မိတ်ဆက်ကို [ဤနေရာတွင်](https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Client-side_web_APIs/Introduction) ဖတ်ရှုနိုင်ပါတယ်။ အများသုံးနိုင်သော [Public API အများအပြား](https://github.com/toddmotto/public-apis) ရှိကြပါတယ်။ Web API များသည် အကြောင်းရင်း အမြောက်အမြားအတွက် အသုံးဝင်လှပါတယ်-

- **Retrieval (အချက်အလက်ရယူခြင်း)** - Web API များသည် မြေပုံများ၊ မိုးလေဝသ အခြေအနေ သို့မဟုတ် သင့် Public IP Address ကဲ့သို့သော အချက်အလက်များကို အလွယ်တကူ ပံ့ပိုးပေးနိုင်ပါတယ်။ ဥပမာအားဖြင့် `curl ipinfo.io` သည် သင့် Public IP၊ တိုင်းဒေသကြီး၊ တည်နေရာ စသည်တို့နှင့် ပတ်သက်သည့် အသေးစိတ် အချက်အလက်အချို့ ပါဝင်သော JSON object တစ်ခုကို ပြန်ပေးမည် ဖြစ်သည်။ စနစ်တကျ Parse ပြုလုပ်ခြင်းဖြင့် ဤ Tool များကို Command Line Tool များနှင့်ပင် ချိတ်ဆက်နိုင်ပါတယ်။ အောက်ပါ Bash function သည် Google ၏ Autocompletion API နှင့် ချိတ်ဆက်ပြီး ပထမဆုံး ကိုက်ညီမှု ၁၀ ခုကို ပြန်လည် ထုတ်ပေးပါသည်။

```
function c() {
    url='https://www.google.com/complete/search?client=hp&hl=en&xhr=t'
    # NB: user-agent must be specified to get back UTF-8 data!
    curl -H 'user-agent: Mozilla/5.0' -sSG --data-urlencode "q=$*" "$url" |
        jq -r ".[1][][0]" |
        sed 's,</\?b>,,g'
}
```

- **Interaction (တုံ့ပြန်ဆောင်ရွက်ခြင်း)** - Web API endpoint များကို လုပ်ဆောင်ချက်များ စတင်ရန် (trigger action) လည်း အသုံးပြုနိုင်ပါတယ်။ ၎င်းတို့သည် သက်ဆိုင်ရာ ဝန်ဆောင်မှုမှတစ်ဆင့် ရယူနိုင်သော Authentication Token တစ်ခုခု လိုအပ်လေ့ရှိပါတယ်။ ဥပမာ အောက်ပါ command ကို သုံးစွဲခြင်း

ဖြင့် `curl -X POST -H 'Content-type: application/json' --data '{"text":"Hello, World!"}' "https://hooks.slack.com/services/$SLACK_TOKEN"` သည် Channel တစ်ခုအတွင်း သို့ `Hello, World!` ဟူသော မက်ဆေ့ဂျ်ကို ပေးပို့ပေးမည် ဖြစ်သည်။

- **Piping (ချိတ်ဆက် အလုပ်လုပ်စေခြင်း)** - Web API ရှိသည့် ဝန်ဆောင်မှု အချို့သည် အတော်ပင် လူကြိုက်များသောကြောင့် အများသုံး Web API များကို အချင်းချင်း ချိတ်ဆက်ပေးခြင်း (gluing) များကို Server တွင် တစ်ပါတည်း ထည့်သွင်း အကောင်အထည်ဖော် ပံ့ပိုးပေးထားပြီး ဖြစ်ပါတယ်။ [If This Then That](https://ifttt.com/) (https://ifttt.com/) နှင့် [Zapier](https://zapier.com/) (https://zapier.com/) ကဲ့သို့သော ဝန်ဆောင်မှုများမှာ ထိုသို့ ပြုလုပ်ပေးထားခြင်း ဖြစ်ပါတယ်။

Web Automation (ဝက်ဘ်ဆိုက် လုပ်ဆောင်ချက်များကို အလိုအလျောက် ခိုင်းစေခြင်း)

အချို့သော အခြေအနေများတွင် Web API များသည် မလုံလောက်ပါ။ အကယ်၍ ဖတ်ရှုရန်သာ လိုအပ်ပါက `pup` ကဲ့သို့သော HTML Parser သို့မဟုတ် Python ၏ BeautifulSoup ကဲ့သို့သော Library များကို အသုံးပြုနိုင်ပါတယ်။ သို့သော် တုံ့ပြန်ဆောင်ရွက်မှု သို့မဟုတ် JavaScript အကောင်အထည်ဖော်မှု လိုအပ်ပါက ထိုနည်းလမ်းများသည် လုံလောက်မှု မရှိတော့ပါ။

- **WebDriver** - ဥပမာ အောက်ပါ Script သည် ဝက်ဘ်ဆိုက် ရိုက်ထည့်သည့် အပြုအမူကို တုပ (simulate) ပြီး သတ်မှတ်ထားသော URL ကို Wayback Machine အသုံးပြု၍ သိမ်းဆည်းပေးမည် ဖြစ်သည်-

```
from selenium.webdriver import Firefox
from selenium.webdriver.common.keys import Keys


def snapshot_wayback(driver, url):

    driver.get("https://web.archive.org/")
    elem = driver.find_element_by_class_name('web-save-url-input')
    elem.clear()
    elem.send_keys(url)
    elem.send_keys(Keys.RETURN)
    driver.close()


driver = Firefox()
url = 'https://hacker-tools.github.io'
snapshot_wayback(driver, url)
```

Exercises (လေ့ကျင့်ခန်းများ)

1. သင်၏ Web Browser တွင် မကြာခဏ အသုံးပြုလေ့ရှိသော Keyword Search Engine တစ်ခုကို ပြင်ဆင်ကြည့်ပါ။
2. ဖော်ပြခဲ့သော Extension များကို ထည့်သွင်းပါ။ ဝက်ဘ်ဆိုက်တစ်ခုအတွက် uBlock Origin/Privacy Badger ကို မည်သို့ ပိတ်နိုင်မည်နည်းဆိုသည်ကို လေ့လာပါ။ မည်သည့် ကွာခြားချက်များကို မြင်တွေ့ရပါသနည်း။ YouTube ကဲ့သို့ ကြော်ငြာများပြားသည့် ဝက်ဘ်ဆိုက်တစ်ခုတွင် စမ်းသပ်ကြည့်ပါ။
3. Stylus ကို ထည့်သွင်းပြီး ပေးထားသော CSS ကို အသုံးပြု၍ သင်တန်း ဝက်ဘ်ဆိုက်အတွက် Custom Style တစ်ခု ရေးသားပါ။ အသုံးများသော ပရိုဂရမ်းမင်း သင်္ကေတများမှာ `= == === >= => ++ /= ~=` ဖြစ်ကြသည်။ Font ကို Fira Code သို့ ပြောင်းလိုက်သည့်အခါ ၎င်းတို့တွင် မည်သို့ ပြောင်းလဲသွားသနည်း။ ပိုမိုသိရှိလိုပါက Programming Font Ligatures အကြောင်း ရှာဖွေကြည့်ပါ။
4. သင့်မြို့/ဒေသ၏ မိုးလေဝသ အခြေအနေကို ရယူနိုင်သည့် Web API တစ်ခုကို ရှာဖွေပါ။
5. သင်၏ Browser တွင် မကြာခဏ ပြုလုပ်ရသည့် ထပ်တလဲလဲ အလုပ်များကို အလိုအလျောက် ပြုလုပ်နိုင်ရန် **Selenium** (https://www.selenium.dev/documentation/) ကဲ့သို့သော WebDriver Software တစ်ခုကို အသုံးပြုကြည့်ပါ။

🔗 External Links & References (ပြင်ပ လင့်ခ်များနှင့် ကိုးကားချက်များ)

Resource / Description	URL	Micro QR
1. Google	https://ahrefs.com/blog/google-advanced-search-operators/	
2. DuckDuckGo	https://duck.co/help/results/syntax	
3. Chrome	https://chrome.google.com/webstore/detail/ublock-origin/cjpalhdlnbpafiamejdnhcphjbkeiagm	
4. Firefox	https://addons.mozilla.org/en-US/firefox/addon/ublock-origin/	
5. အင်တာနက်ပေါ်ရှိ	https://github.com/gorhill/uBlock/wiki/Filter-lists-from-around-the-web	
6. Privacy Badger	https://privacybadger.org/	
7. HTTPS everywhere	https://www.eff.org/https-everywhere	
8. ဤနေရာတွင်	https://www.privacytools.io/privacy-browser-addons/	
9. Stylus	https://github.com/openstyles/stylus	
10. Firefox	https://addons.mozilla.org/en-US/firefox/addon/styl-us/	
11. Chrome	https://chrome.google.com/webstore/detail/stylus/clngdbkpkpeebahjckkjfobafhncgmne?hl=en	
12. userstyles.org	https://userstyles.org/	
13. ဤနေရာတွင်	https://arstechnica.com/information-technology/2018/07/stylish-extension-with-2m-downloads-banished-for-tracking-every-site-visit/	
14. Tampermonkey	https://tampermonkey.net/	
15. OpenUserJS	https://openuserjs.org/	

Resource / Description	URL	Micro QR
16. Greasy Fork	https://greasyfork.org/en	
17. ဤနေရာတွင်	https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Client-side_web_APIs/Introduction	
18. Public API အများအပြား	https://github.com/toddmotto/public-apis	
19. If This Then That	https://ifttt.com/	
20. Zapier	https://zapier.com/	
21. Selenium	https://www.selenium.dev/documentation/	

လိုင်စင်နှင့် မူပိုင်ခွင့် (License)

မူပိုင်ခွင့် လိုင်စင် (License)

ဤသင်တန်းရှိ ဝဘ်ဆိုက် source code၊ သင်ခန်းစာ မှတ်တမ်းများ၊ လေ့ကျင့်ခန်းများနှင့် သင်ခန်းစာ ဗီဒီယို များ အပါအဝင် အကြောင်းအရာ အားလုံးကို CC BY-NC-SA 4.0 လိုင်စင်ဖြင့် ထုတ်ဝေထားပါသည်။

ဤသည်မှာ သင်၏ အခွင့်အရေးများ ဖြစ်ကြပါသည် - - **Share** — မည်သည့် မီဒီယာ သို့မဟုတ် မူဘောင် ဖြင့် မဆို အကြောင်းအရာများကို ကူးယူ ပြန်လည် ဖြန့်ဝေနိုင်ပါသည်။ - **Adapt** — အကြောင်းအရာများကို ပြုပြင် ပြောင်းလဲခြင်း၊ တည်ဆောက်ခြင်းများ ပြုလုပ်နိုင်ပါသည်။

အောက်ပါ စည်းကမ်းချက်များနှင့် အညီ ပြုလုပ်ရမည် -

- **Attribution** — သင်သည် မူရင်း ဖန်တီးသူအား သင့်တော်သော အသိအမှတ်ပြုမှု ပေးရမည်၊ လိုင်စင်သို့ လင့်ခ် ချိတ်ဆက်ပေးရမည်၊ ပြောင်းလဲမှုများ ပြုလုပ်ထားပါက ညွှန်းဆိုပေးရမည်။
- **NonCommercial** — ဤ အကြောင်းအရာများကို စီးပွားရေး ရည်ရွယ်ချက်ဖြင့် အသုံးပြုပိုင်ခွင့် မရှိပါ။
- **ShareAlike** — အကယ်၍ သင်သည် ဤ အကြောင်းအရာများကို ပြုပြင် ပြောင်းလဲပါက၊ သင်၏ ပံ့ပိုးမှု များကို မူရင်း လိုင်စင်အတိုင်း အတိအကျ ပြန်လည် ဖြန့်ဝေရမည်။

ဤသည်မှာ မူပိုင်ခွင့် Legal Code ၏ လူနားလည်လွယ်သော အတိုချုပ် ဖြစ်ပါသည်။

Contribution guidelines

ကျွန်ုပ်တို့၏ GitHub repo တွင် Issues နှင့် Pull Requests တင်သွင်းခြင်းဖြင့် သင်ခန်းစာ အကြောင်းအရာ များအတွက် ပြင်ဆင်ချက်များနှင့် အကြံပြုချက်များကို တင်သွင်းနိုင်ပါသည်။ ၎င်းတွင် ဗီဒီယို စာတန်းထိုး များလည်း ပါဝင်ပါသည် (ဒီမှာ ကြည့်ပါ)။

Translation guidelines

မူပိုင်ခွင့် စည်းကမ်းချက်များကို လိုက်နာသရွေ့ သင်ခန်းစာ မှတ်တမ်းများနှင့် လေ့ကျင့်ခန်းများကို လွတ်လပ် စွာ ဘာသာပြန်ဆိုနိုင်ပါသည်။ သင်၏ ဘာသာပြန်ဆိုမှုသည် သင်တန်း ပုံစံအတိုင်း ဖြစ်ပါက၊ ကျွန်ုပ်တို့၏ စာမျက်နှာမှတဆင့် သင်၏ ဘာသာပြန် မူကွဲကို လင့်ခ် ချိတ်ဆက်ပေးနိုင်ရန် ကျွန်ုပ်တို့ထံ ဆက်သွယ်ပါ။

ဗီဒီယို စာတန်းထိုးများ ဘာသာပြန်ဆိုရန်အတွက် YouTube တွင် Community contribution အဖြစ် တင် သွင်းပေးပါ။

MIT CSIL • BURMESE EBOOK EDITION

The Missing Semester of Your CS Education

(2019 Burmese Edition)

ORIGINAL COURSE

Originally designed and taught at Massachusetts Institute of Technology (MIT) by **Anish Athalye, Jon Gjengset, and Jose Javier Gonzalez Ortiz**.

Original Website: https://missing.csail.mit.edu/

MY LOCALIZATION & PUBLICATION

Burmese translation and digital eBook publication maintained by **FOSS Myanmar** (fossmyanmar.org) and **Vibe Code Tours** (vibecode.tours).

Burmese Website: https://missing-semester-my.github.io/

Scan to Visit Burmese Course Portal:

https://missing-semester-my.github.io/