INTRO TO LINUX SERVER & CCR

SESSION 04

By: Ningji Wei

November 15, 2018

University at Buffalo The State University of New York

Well Done!

Basics:
know how to finish common tasks in CCR

- ☑ Connect to CCR
- ☑ Filesystem Operations
- ☑ CCR Operations
- ☑ File Transfer

Contents:

1. Introduction to Linux Server & CCR
2. Linux Basics (Commands & Tricks)
3. Run Jobs in CCR
4. Intro to Advanced Session
5. Advanced Tools

Linux:

Linux:
Is About Free and Open

Linux:
Is About Free and Open

transition of mentality

Linux:
Is About Free and Open

How can I learn stuff to use certain interface / application / workflow?

transition of mentality

Linux:
Is About Free and Open

How can I learn stuff to use certain interface / application / workflow?

transition of mentality

How can I tweak the computer to make it adapt to what I want?

Linux:
Is About Free and Open

How can I learn stuff to use certain interface / application / workflow?

transition of mentality

How can I tweak the computer to make it adapt to what I want?

To achieve this:

Linux:
Is About Free and Open

How can I learn stuff to use certain interface / application / workflow?

transition of mentality

How can I tweak the computer to make it adapt to what I want?

To achieve this:

- Choices on apps

Linux:
Is About Free and Open

How can I learn stuff to use certain interface / application / workflow?

transition of mentality

How can I tweak the computer to make it adapt to what I want?

To achieve this:

- Choices on apps
- Almost everything is configurable

Linux:
Is About Free and Open

How can I learn stuff to use certain interface / application / workflow?

transition of mentality

How can I tweak the computer to make it adapt to what I want?

To achieve this:

- Choices on apps
- Almost everything is configurable
- Almost everything is open source

Linux:
Is About Free and Open

How can I learn stuff to use certain interface / application / workflow?

transition of mentality

How can I tweak the computer to make it adapt to what I want?

To achieve this:

- Choices on apps
- Almost everything is configurable
- Almost everything is open source
- Code it yourself (sometimes just one line of code)

Linux:
Is About Free and Open

How can I learn stuff to use certain interface / application / workflow?

transition of mentality

How can I tweak the computer to make it adapt to what I want?

To achieve this:

- Choices on apps
- Almost everything is configurable
- Almost everything is open source
- Code it yourself (sometimes just one line of code)

MOST IMPORTANT:

Linux:
Is About Free and Open

How can I learn stuff to use certain interface / application / workflow?

transition of mentality

How can I tweak the computer to make it adapt to what I want?

To achieve this:

- Choices on apps
- Almost everything is configurable
- Almost everything is open source
- Code it yourself (sometimes just one line of code)

MOST IMPORTANT:

- Ask yourself, what you like, what you want

Linux:
Is About Free and Open

How can I learn stuff to use certain interface / application / workflow?

transition of mentality

How can I tweak the computer to make it adapt to what I want?

To achieve this:

- Choices on apps
- Almost everything is configurable
- Almost everything is open source
- Code it yourself (sometimes just one line of code)

MOST IMPORTANT:

- Ask yourself, what you like, what you want
- Search it online

Linux:
Is About Free and Open

How can I learn stuff to use certain interface / application / workflow?

transition of mentality

How can I tweak the computer to make it adapt to what I want?

To achieve this:

- Choices on apps
- Almost everything is configurable
- Almost everything is open source
- Code it yourself (sometimes just one line of code)

MOST IMPORTANT:

- Ask yourself, what you like, what you want
- Search it online
- Practice it on your own!

Let’s Connect to CCR Again

Let’s Connect to CCR Again

What I want:

Let's Connect to CCR Again

What I want:

- Connect to CCR with one simple command: $ ccr

Let's Connect to CCR Again

What I want:

- Connect to CCR with one simple command: $ ccr
- No need to input password, but also keep the communication safe

Let's Connect to CCR Again

What I want:

- Connect to CCR with one simple command: $ ccr
- No need to input password, but also keep the communication safe

How to achieve:

Let's Connect to CCR Again

What I want:

- Connect to CCR with one simple command: $ ccr
- No need to input password, but also keep the communication safe

How to achieve:

- Create an one line script

Let's Connect to CCR Again

What I want:

- Connect to CCR with one simple command: $ ccr
- No need to input password, but also keep the communication safe

How to achieve:

- Create an one line script
- Make it executable: $ chmod +x ccr

Let's Connect to CCR Again

What I want:

- Connect to CCR with one simple command: $ ccr
- No need to input password, but also keep the communication safe

How to achieve:

- Create an one line script
- Make it executable: $ chmod +x ccr
- Add the folder to path: $ export PATH=$PATH:~/bin

Let's Connect to CCR Again

What I want:

- Connect to CCR with one simple command: $ ccr
- No need to input password, but also keep the communication safe

How to achieve:

- Create an one line script
- Make it executable: $ chmod +x ccr
- Add the folder to path: $ export PATH=$PATH:~/bin
- ** Use ssh-key to generate key pairs, for safe but easy login

Let's Connect to CCR Again

What I want:

- Connect to CCR with one simple command: $ ccr
- No need to input password, but also keep the communication safe

How to achieve:

- Create an one line script
- Make it executable: $ chmod +x ccr
- Add the folder to path: $ export PATH=$PATH:~/bin
- ** Use ssh-key to generate key pairs, for safe but easy login (https://www.debian.org/devel/passwordlessssh)

Let's Connect to CCR Again

What I want:

- Connect to CCR with one simple command: $ ccr
- No need to input password, but also keep the communication safe

How to achieve:

- Create an one line script
- Make it executable: $ chmod +x ccr
- Add the folder to path: $ export PATH=$PATH:~/bin
- ** Use ssh-key to generate key pairs, for safe but easy login (https://www.debian.org/devel/passwordlessssh)

It is not difficult at all!

Let's Connect to CCR Again

What I want:

- Connect to CCR with one simple command: $ ccr
- No need to input password, but also keep the communication safe

How to achieve:

- Create an one line script
- Make it executable: $ chmod +x ccr
- Add the folder to path: $ export PATH=$PATH:~/bin
- ** Use ssh-key to generate key pairs, for safe but easy login (https://www.debian.org/devel/passwordlessssh)

It is not difficult at all!

The most important is your willingness to practice your idea!

Let's Connect to CCR Again

What I want:

- Connect to CCR with one simple command: $ ccr
- No need to input password, but also keep the communication safe

How to achieve:

- Create an one line script # Shell script & Shebang
- Make it executable: $ chmod +x ccr
- Add the folder to path: $ export PATH=$PATH:~/bin
- ** Use ssh-key to generate key pairs, for safe but easy login (https://www.debian.org/devel/passwordlessssh)

It is not difficult at all!

The most important is your willingness to practice your idea!

Let's Connect to CCR Again

What I want:

- Connect to CCR with one simple command: $ ccr
- No need to input password, but also keep the communication safe

How to achieve:

- Create an one line script # Shell script & Shebang
- Make it executable: $ chmod +x ccr # File permission
- Add the folder to path: $ export PATH=$PATH:~/bin
- ** Use ssh-key to generate key pairs, for safe but easy login (https://www.debian.org/devel/passwordlessssh)

It is not difficult at all!

The most important is your willingness to practice your idea!

Let's Connect to CCR Again

What I want:

- Connect to CCR with one simple command: $ ccr
- No need to input password, but also keep the communication safe

How to achieve:

- Create an one line script # Shell script & Shebang
- Make it executable: $ chmod +x ccr # File permission
- Add the folder to path: $ export PATH=$PATH:~/bin # Environment Variables
- ** Use ssh-key to generate key pairs, for safe but easy login (https://www.debian.org/devel/passwordlessssh)

It is not difficult at all!

The most important is your willingness to practice your idea!

Let's Connect to CCR Again

What I want:

- Connect to CCR with one simple command: $ ccr
- No need to input password, but also keep the communication safe

How to achieve:

- Create an one line script # Shell script & Shebang
- Make it executable: $ chmod +x ccr # File permission
- Add the folder to path: $ export PATH=$PATH:~/bin # Environment Variables
- ** Use ssh-key to generate key pairs, for safe but easy login (https://www.debian.org/devel/passwordlessssh) # Asymmetric cryptography

It is not difficult at all!

The most important is your willingness to practice your idea!

Contents:

1. Introduction to Linux Server & CCR
2. Linux Basics (Commands & Tricks)
3. Run Jobs in CCR
4. Intro to Advanced Session
5. Advanced Tools

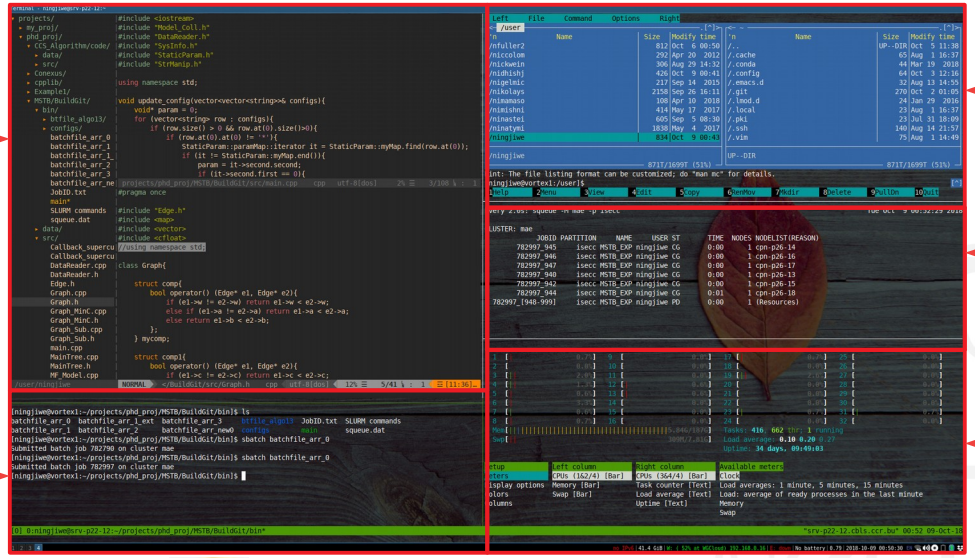
Remember Our Goal

Remember Our Goal

After the workshop

Coding IDE

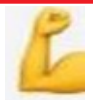
Debug



File manager

Job monitor

Resources monitor



call me Linux server ninja warrior

Remember Our Goal

After the workshop

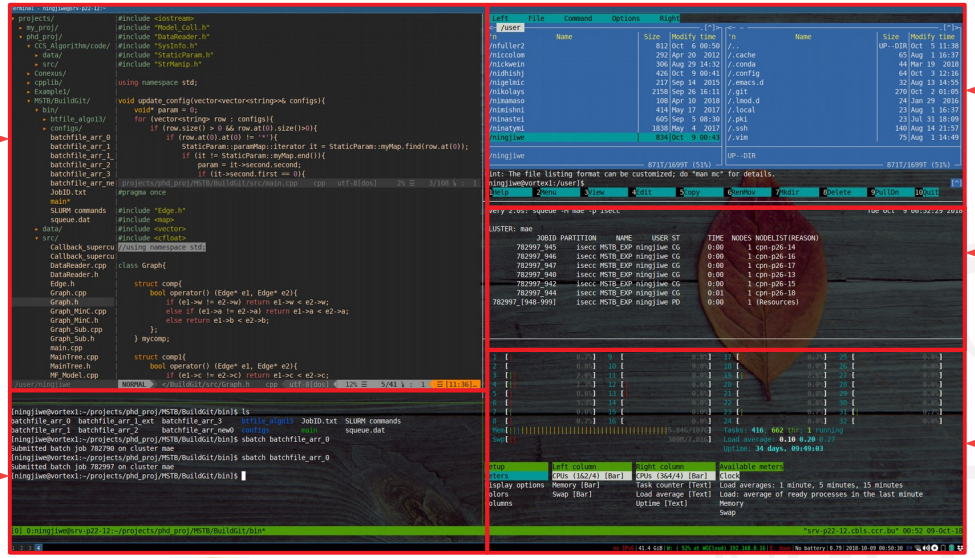
Coding IDE

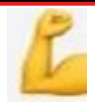
File manager

Job monitor

Resources monitor

Debug





call me Linux server ninja warrior

Toolset:

- Resources monitor: Htop
- File Manger: Midnight Commander
- Multitasking: Tmux
- Text Editor: Vim
- Console Debugger: gdb, jdb, pdb ...

System Monitoring App: Htop

System Monitoring App: Htop

Functionalities & Features:

System Monitoring App: Htop

Functionalities & Features:

- Open by $ htop

System Monitoring App: Htop

Functionalities & Features:

- Open by $ htop
- Read off different system status

System Monitoring App: Htop

Functionalities & Features:

- Open by $ htop
- Read off different system status
- Four regions of the interface

System Monitoring App: Htop

Functionalities & Features:

- Open by $ htop
- Read off different system status
- Four regions of the interface
- Clickable

System Monitoring App: Htop

Functionalities & Features:

- Open by $ htop
- Read off different system status
- Four regions of the interface
- Clickable
- Kill tasks

System Monitoring App: Htop

Functionalities & Features:

- Open by $ htop
- Read off different system status
- Four regions of the interface
- Clickable
- Kill tasks
- Visualize system info in different ways (show timer)

System Monitoring App: Htop

Functionalities & Features:

- Open by $ htop
- Read off different system status
- Four regions of the interface
- Clickable
- Kill tasks
- Visualize system info in different ways (show timer)

Tips for learning new app:

- Search “AppName cheat sheet”, “AppName user guide”...
- Just learn several most useful things at a time

System Monitoring App: Htop

Functionalities & Features:

- Open by $ htop
- Read off different system status
- Four regions of the interface
- Clickable
- Kill tasks
- Visualize system info in different ways (show timer)

Tips for learning new app:

- Search “AppName cheat sheet”, “AppName user guide”...
- Just learn several most useful things at a time

Cheat sheet for htop:

- https://www.maketecheasier.com/power-user-guide-htop/

File Manager: Midnight Commander

File Manager: Midnight Commander

Functionalities & Features:

File Manager: Midnight Commander

Functionalities & Features:

- Open by $ mc

File Manager: Midnight Commander

Functionalities & Features:

- Open by $ mc
- View, create, copy, move, delete folders/files

File Manager: Midnight Commander

Functionalities & Features:

- Open by $ mc
- View, create, copy, move, delete folders/files
- Clickable

File Manager: Midnight Commander

Functionalities & Features:

- Open by $ mc
- View, create, copy, move, delete folders/files
- Clickable
- Change layout

File Manager: Midnight Commander

Functionalities & Features:

- Open by $ mc
- View, create, copy, move, delete folders/files
- Clickable
- Change layout
- Redefine hotkeys

File Manager: Midnight Commander

Functionalities & Features:

- Open by $ mc
- View, create, copy, move, delete folders/files
- Clickable
- Change layout
- Redefine hotkeys

Useful keyboard shortcuts:

- C-o: switch between mc and console
- Tab: jump between panels
- C-t/Insert: select/unselect item
- Alt-.: show/hide hidden files
- Function keys

File Manager: Midnight Commander

Functionalities & Features:

- Open by $ mc
- View, create, copy, move, delete folders/files
- Clickable
- Change layout
- Redefine hotkeys

Useful keyboard shortcuts:

- C-o: switch between mc and console
- Tab: jump between panels
- C-t/Insert: select/unselect item
- Alt-.: show/hide hidden files
- Function keys

Cheat sheet for mc:

- https://www.cheatography.com/hank/cheat-sheets/midnight-commander/

Things We Can Do Now

Things We Can Do Now

- Command line (bash)

Things We Can Do Now

- Command line (bash)
- Edit files (nano)

Things We Can Do Now

- Command line (bash)
- Edit files (nano)
- Monitor jobs (with watch command)

Things We Can Do Now

- Command line (bash)
- Edit files (nano)
- Monitor jobs (with watch command)
- Check system status (htop)

Things We Can Do Now

- Command line (bash)
- Edit files (nano)
- Monitor jobs (with watch command)
- Check system status (htop)
- Manage files (midnight commander)

Things We Can Do Now

- Command line (bash)
- Edit files (nano)
- Monitor jobs (with watch command)
- Check system status (htop)
- Manage files (midnight commander)

We can only do these things one by one!

Things We Can Do Now

- Command line (bash)
- Edit files (nano)
- Monitor jobs (with watch command)
- Check system status (htop)
- Manage files (midnight commander)

We can only do these things one by one!

What We Want?

Things We Can Do Now

- Command line (bash)
- Edit files (nano)
- Monitor jobs (with watch command)
- Check system status (htop)
- Manage files (midnight commander)

We can only do these things one by one!

What We Want?

- Multiple panes (in a window)

Things We Can Do Now

- Command line (bash)
- Edit files (nano)
- Monitor jobs (with watch command)
- Check system status (htop)
- Manage files (midnight commander)

We can only do these things one by one!

What We Want?

- Multiple panes (in a window)
- Maximize/minimize panes

Things We Can Do Now

- Command line (bash)
- Edit files (nano)
- Monitor jobs (with watch command)
- Check system status (htop)
- Manage files (midnight commander)

We can only do these things one by one!

What We Want?

- Multiple panes (in a window)
- Maximize/minimize panes
- Copy/paste texts between panes

Things We Can Do Now

- Command line (bash)
- Edit files (nano)
- Monitor jobs (with watch command)
- Check system status (htop)
- Manage files (midnight commander)

We can only do these things one by one!

What We Want?

- Multiple panes (in a window)
- Maximize/minimize panes
- Copy/paste texts between panes
- Multiple windows (in a session)

Things We Can Do Now

- Command line (bash)
- Edit files (nano)
- Monitor jobs (with watch command)
- Check system status (htop)
- Manage files (midnight commander)

We can only do these things one by one!

What We Want?

- Multiple panes (in a window)
- Maximize/minimize panes
- Copy/paste texts between panes
- Multiple windows (in a session)
- Multiple sessions

Things We Can Do Now

- Command line (bash)
- Edit files (nano)
- Monitor jobs (with watch command)
- Check system status (htop)
- Manage files (midnight commander)

We can only do these things one by one!

What We Want?

- Multiple panes (in a window)
- Maximize/minimize panes
- Copy/paste texts between panes
- Multiple windows (in a session)
- Multiple sessions
- Recoverable sessions

Terminal Multiplier: Tmux

Terminal Multiplier: Tmux

Functionalities & Features:

- Open by $ tmux
- Create sessions, windows and panes.

Terminal Multiplier: Tmux

Functionalities & Features:

- Open by $ tmux
- Create sessions, windows and panes.

Session Operations

• Start new session:	$ tmux
• Detach session:	C-b d
• Start new session with name:	$ tmux new -s NAME
• Attach recent session:	$ tmux a
• Attach specific session:	$ tmux a -t NAME
• List sessions:	$ tmux ls
• Kill specific session:	$ tmux kill-ses -t NAME
• Kill all sessions but the current:	$ tmux kill-ses -a
• Kill all sessions:	$ tmux kill-server

Terminal Multiplier: Tmux

Functionalities & Features:

- Open by $ tmux
- Create sessions, windows and panes.

Session Operations

• Start new session:	$ tmux
• Detach session:	C-b d
• Start new session with name:	$ tmux new -s NAME
• Attach recent session:	$ tmux a
• Attach specific session:	$ tmux a -t NAME
• List sessions:	$ tmux ls
• Kill specific session:	$ tmux kill-ses -t NAME
• Kill all sessions but the current:	$ tmux kill-ses -a
• Kill all sessions:	$ tmux kill-server

In Session: Command Prefix

- Prefix before all command: C-b

Terminal Multiplier: Tmux

Terminal Multiplier: Tmux

Pane Operations (With Command Prefix)

- Vertical split: %
- Horizontal split: ”
- Kill pane: x
- Jump between panes: arrow keys
- Move pane left: {
- Move pane right: }
- Zoom in/out current pane: z
- Auto change layouts: space
- Resize pane: hold prefix key + arrow keys

Terminal Multiplier: Tmux

Pane Operations (With Command Prefix)

- Vertical split: %
- Horizontal split: ”
- Kill pane: x
- Jump between panes: arrow keys
- Move pane left: {
- Move pane right: }
- Zoom in/out current pane: z
- Auto change layouts: space
- Resize pane: hold prefix key + arrow keys

Remember Linux is all about what you want!

Terminal Multiplier: Tmux

Pane Operations (With Command Prefix)

- Vertical split: %
- Horizontal split: ”
- Kill pane: x
- Jump between panes: arrow keys
- Move pane left: {
- Move pane right: }
- Zoom in/out current pane: z
- Auto change layouts: space
- Resize pane: hold prefix key + arrow keys

Remember Linux is all about what you want!

Auto-start Session with Layouts and Apps

Terminal Multiplier: Tmux

Pane Operations (With Command Prefix)

- Vertical split: %
- Horizontal split: ”
- Kill pane: x
- Jump between panes: arrow keys
- Move pane left: {
- Move pane right: }
- Zoom in/out current pane: z
- Auto change layouts: space
- Resize pane: hold prefix key + arrow keys

Remember Linux is all about what you want!

Auto-start Session with Layouts and Apps

- Write a short bash script with handful lines

Terminal Multiplier: Tmux

Pane Operations (With Command Prefix)

- Vertical split: %
- Horizontal split: ”
- Kill pane: x
- Jump between panes: arrow keys
- Move pane left: {
- Move pane right: }
- Zoom in/out current pane: z
- Auto change layouts: space
- Resize pane: hold prefix key + arrow keys

Remember Linux is all about what you want!

Auto-start Session with Layouts and Apps

- Write a short bash script with handful lines
- https://stackoverflow.com/questions/5609192/how-to-set-up-tmux-so-that-it-starts-up-with-specified-windows-opened

Terminal Multiplier: Tmux

Terminal Multiplier: Tmux

Window Operations (With Command Prefix)

- Create window: c
- List windows: w
- Next window: n
- Previous window: p
- Kill window: &
- Go to window by number: 0 … 9

Terminal Multiplier: Tmux

Window Operations (With Command Prefix)

- Create window: c
- List windows: w
- Next window: n
- Previous window: p
- Kill window: &
- Go to window by number: 0 … 9

Clipboard (Copy & Paste between panes)

Cheat sheet for tmux:

Terminal Multiplier: Tmux

Window Operations (With Command Prefix)

- Create window: c
- List windows: w
- Next window: n
- Previous window: p
- Kill window: &
- Go to window by number: 0 … 9

Clipboard (Copy & Paste between panes)

Cheat sheet for tmux:

- https://gist.github.com/MohamedAlaa/2961058

Terminal Multiplier: Tmux

Window Operations (With Command Prefix)

- Create window: c
- List windows: w
- Next window: n
- Previous window: p
- Kill window: &
- Go to window by number: 0 … 9

Clipboard (Copy & Paste between panes)

Cheat sheet for tmux:

- https://gist.github.com/MohamedAlaa/2961058
- http://tmuxcheatsheet.com/

Text Editor / All Around IDE: Vi

Text Editor / All Around IDE: Vi

Vi represents a class of text editors:

Text Editor / All Around IDE: Vi

Vi represents a class of text editors:

- Vi: original editor, almost in every Linux environment

Text Editor / All Around IDE: Vi

Vi represents a class of text editors:

- Vi: original editor, almost in every Linux environment
- Vim: vi improved

Text Editor / All Around IDE: Vi

Vi represents a class of text editors:

- Vi: original editor, almost in every Linux environment
- Vim: vi improved
- Gvim: vim with GUI

Text Editor / All Around IDE: Vi

Vi represents a class of text editors:

- Vi: original editor, almost in every Linux environment
- Vim: vi improved
- Gvim: vim with GUI
- Neovim: the future of vim (according to their website)

Text Editor / All Around IDE: Vi

Vi represents a class of text editors:

- Vi: original editor, almost in every Linux environment
- Vim: vi improved
- Gvim: vim with GUI
- Neovim: the future of vim (according to their website)
- You can start with any of them, and you'll know how to use the rest naturally

Text Editor / All Around IDE: Vi

Vi represents a class of text editors:

- Vi: original editor, almost in every Linux environment
- Vim: vi improved
- Gvim: vim with GUI
- Neovim: the future of vim (according to their website)
- You can start with any of them, and you'll know how to use the rest naturally
- Vi and vim are available in CCR.

Text Editor / All Around IDE: Vi

Vi represents a class of text editors:

- Vi: original editor, almost in every Linux environment
- Vim: vi improved
- Gvim: vim with GUI
- Neovim: the future of vim (according to their website)
- You can start with any of them, and you'll know how to use the rest naturally
- Vi and vim are available in CCR.

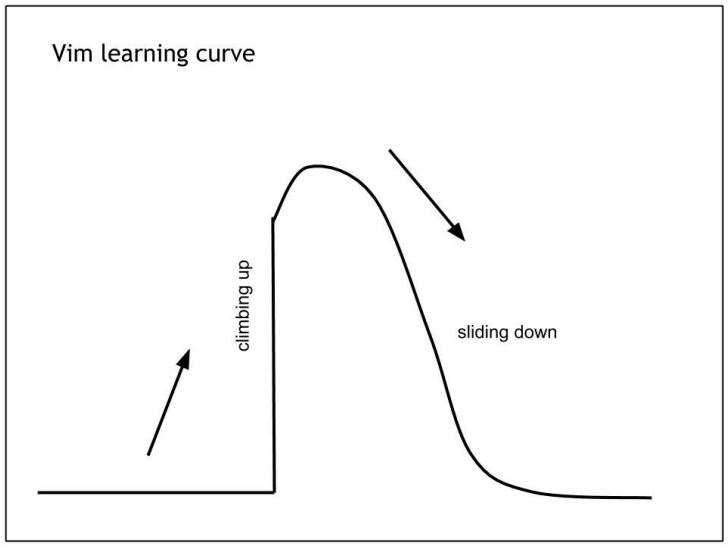
Vim is not easy to get started with (same with Emacs),

Text Editor / All Around IDE: Vi

Vi represents a class of text editors:

- Vi: original editor, almost in every Linux environment
- Vim: vi improved
- Gvim: vim with GUI
- Neovim: the future of vim (according to their website)
- You can start with any of them, and you'll know how to use the rest naturally
- Vi and vim are available in CCR.

Vim is not easy to get started with (same with Emacs),



Courtesy: https://pascalprecht.github.io/2014/03/18/why-i-use-vim/

Text Editor / All Around IDE: Vi

Text Editor / All Around IDE: Vi

But it is an app that reward you a lot along the way

Text Editor / All Around IDE: Vi

But it is an app that reward you a lot along the way

- Initial release: 1976

Text Editor / All Around IDE: Vi

But it is an app that reward you a lot along the way

- Initial release: 1976
- Still on the top of the list in almost all comparisons of text editors
  https://www.slant.co/topics/10214/~linux-text-editors

Text Editor / All Around IDE: Vi

But it is an app that reward you a lot along the way

- Initial release: 1976
- Still on the top of the list in almost all comparisons of text editors
  https://www.slant.co/topics/10214/~linux-text-editors
- It is everywhere, all platform.

Text Editor / All Around IDE: Vi

But it is an app that reward you a lot along the way

- Initial release: 1976
- Still on the top of the list in almost all comparisons of text editors https://www.slant.co/topics/10214/~linux-text-editors
- It is everywhere, all platform.
- It is fast and efficient.

Text Editor / All Around IDE: Vi

But it is an app that reward you a lot along the way

- Initial release: 1976
- Still on the top of the list in almost all comparisons of text editors https://www.slant.co/topics/10214/~linux-text-editors
- It is everywhere, all platform.
- It is fast and efficient.
- It is can be a full-featured IDE for almost every language.

Text Editor / All Around IDE: Vi

But it is an app that reward you a lot along the way

- Initial release: 1976
- Still on the top of the list in almost all comparisons of text editors https://www.slant.co/topics/10214/~linux-text-editors
- It is everywhere, all platform.
- It is fast and efficient.
- It is can be a full-featured IDE for almost every language.
- It can be a full-featured IDE for LaTeX.

Text Editor / All Around IDE: Vi

But it is an app that reward you a lot along the way

- Initial release: 1976
- Still on the top of the list in almost all comparisons of text editors https://www.slant.co/topics/10214/~linux-text-editors
- It is everywhere, all platform.
- It is fast and efficient.
- It is can be a full-featured IDE for almost every language.
- It can be a full-featured IDE for LaTeX.
- You can configure it as simple / fancy / efficient as you want.

Text Editor / All Around IDE: Vi

But it is an app that reward you a lot along the way

- Initial release: 1976
- Still on the top of the list in almost all comparisons of text editors https://www.slant.co/topics/10214/~linux-text-editors
- It is everywhere, all platform.
- It is fast and efficient.
- It is can be a full-featured IDE for almost every language.
- It can be a full-featured IDE for LaTeX.
- You can configure it as simple / fancy / efficient as you want.

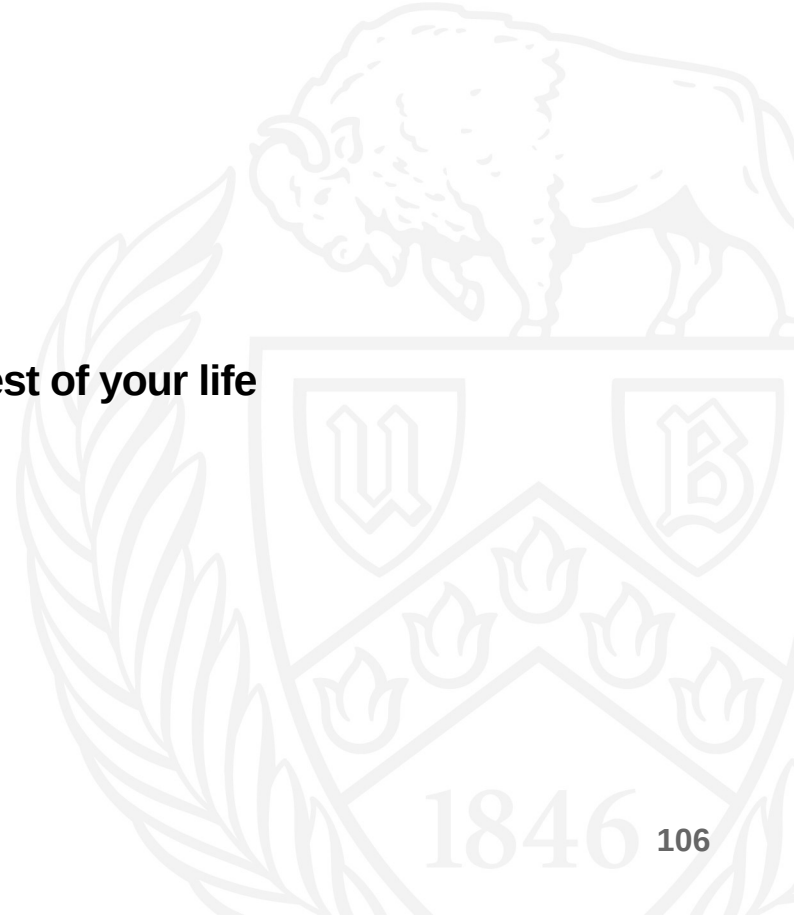
And after certain point,

Text Editor / All Around IDE: Vi

But it is an app that reward you a lot along the way

- Initial release: 1976
- Still on the top of the list in almost all comparisons of text editors
  https://www.slant.co/topics/10214/~linux-text-editors
- It is everywhere, all platform.
- It is fast and efficient.
- It is can be a full-featured IDE for almost every language.
- It can be a full-featured IDE for LaTeX.
- You can configure it as simple / fancy / efficient as you want.

And after certain point,
you just want to use it for every possible task for the rest of your life



Text Editor / All Around IDE: Vi

But it is an app that reward you a lot along the way

- Initial release: 1976
- Still on the top of the list in almost all comparisons of text editors
  https://www.slant.co/topics/10214/~linux-text-editors
- It is everywhere, all platform.
- It is fast and efficient.
- It is can be a full-featured IDE for almost every language.
- It can be a full-featured IDE for LaTeX.
- You can configure it as simple / fancy / efficient as you want.

And after certain point,
you just want to use it for every possible task for the rest of your life

According to Slate, Emacs and Vi are “Among the longest-lived applications of all time.”
The Wikipedia entry on the differences between Emacs and Vi describes the Holy War between the two applications which has raged since the late-1970s.

Text Editor / All Around IDE: Vi

But it is an app that reward you a lot along the way

- Initial release: 1976
- Still on the top of the list in almost all comparisons of text editors https://www.slant.co/topics/10214/~linux-text-editors
- It is everywhere, all platform.
- It is fast and efficient.
- It is can be a full-featured IDE for almost every language.
- It can be a full-featured IDE for LaTeX.
- You can configure it as simple / fancy / efficient as you want.

And after certain point,
you just want to use it for every possible task for the rest of your life

According to Slate, Emacs and Vi are “Among the longest-lived applications of all time.” The Wikipedia entry on the differences between Emacs and Vi describes the Holy War between the two applications which has raged since the late-1970s.

Both editors require learning their respective commands, which means you won’t be able to open them and get right to work. But expert users of either seem to make their editor of choice perform magic, often without even leaving their keyboard’s home row.

https://medium.com/linode-cube/emacs-nano-or-vim-choose-your-terminal-based-text-editor-wisely-8f3826c92a68

What Makes Vim Special: Three Concepts

What Makes Vim Special: Three Concepts

Concept 1: Mode, do different things in different modes

What Makes Vim Special: Three Concepts

Concept 1: Mode, do different things in different modes

- Normal mode: navigation and manipulation of text.

What Makes Vim Special: Three Concepts

Concept 1: Mode, do different things in different modes

- Normal mode: navigation and manipulation of text.
- Insert mode: inserting new text

What Makes Vim Special: Three Concepts

Concept 1: Mode, do different things in different modes

- Normal mode: navigation and manipulation of text.
- Insert mode: inserting new text
- Visual mode: text selection, copy (yank) & paste, operator on blocks

What Makes Vim Special: Three Concepts

Concept 1: Mode, do different things in different modes

- Normal mode: navigation and manipulation of text.
- Insert mode: inserting new text
- Visual mode: text selection, copy (yank) & paste, operator on blocks
- Command-line mode: input command

What Makes Vim Special: Three Concepts

Concept 1: Mode, do different things in different modes

- Normal mode: navigation and manipulation of text.
- Insert mode: inserting new text
- Visual mode: text selection, copy (yank) & paste, operator on blocks
- Command-line mode: input command
- Full introduction: https://guide.freecodecamp.org/vim/modes/

What Makes Vim Special: Three Concepts

Concept 1: Mode, do different things in different modes

- Normal mode: navigation and manipulation of text.
- Insert mode: inserting new text
- Visual mode: text selection, copy (yank) & paste, operator on blocks
- Command-line mode: input command
- Full introduction: https://guide.freecodecamp.org/vim/modes/

Concept 2: Text Objects & Motions, manipulate text naturally (Influence many other apps)

What Makes Vim Special: Three Concepts

Concept 1: Mode, do different things in different modes

- Normal mode: navigation and manipulation of text.
- Insert mode: inserting new text
- Visual mode: text selection, copy (yank) & paste, operator on blocks
- Command-line mode: input command
- Full introduction: https://guide.freecodecamp.org/vim/modes/

Concept 2: Text Objects & Motions, manipulate text naturally (Influence many other apps)

- Command Structure: <command><number><text object or motion>.

What Makes Vim Special: Three Concepts

Concept 1: Mode, do different things in different modes

- Normal mode: navigation and manipulation of text.
- Insert mode: inserting new text
- Visual mode: text selection, copy (yank) & paste, operator on blocks
- Command-line mode: input command
- Full introduction: https://guide.freecodecamp.org/vim/modes/

Concept 2: Text Objects & Motions, manipulate text naturally (Influence many other apps)

- Command Structure: <command><number><text object or motion>.
- Number: how many times to repeat the command.

What Makes Vim Special: Three Concepts

Concept 1: Mode, do different things in different modes

- Normal mode: navigation and manipulation of text.
- Insert mode: inserting new text
- Visual mode: text selection, copy (yank) & paste, operator on blocks
- Command-line mode: input command
- Full introduction: https://guide.freecodecamp.org/vim/modes/

Concept 2: Text Objects & Motions, manipulate text naturally (Influence many other apps)

- Command Structure: <command><number><text object or motion>.
- Number: how many times to repeat the command.
- Command: an operation, d(elete), c(hange), y(ank)

What Makes Vim Special: Three Concepts

Concept 1: Mode, do different things in different modes

- Normal mode: navigation and manipulation of text.
- Insert mode: inserting new text
- Visual mode: text selection, copy (yank) & paste, operator on blocks
- Command-line mode: input command
- Full introduction: https://guide.freecodecamp.org/vim/modes/

Concept 2: Text Objects & Motions, manipulate text naturally (Influence many other apps)

- Command Structure: <command><number><text object or motion>.
- Number: how many times to repeat the command.
- Command: an operation, d(elete), c(hange), y(ank)
- Text object: w(ord), s(entence), p(aragraph), iw(inner word), aw(a word), is, as, ip, ap, i), a).

What Makes Vim Special: Three Concepts

Concept 1: Mode, do different things in different modes

- Normal mode: navigation and manipulation of text.
- Insert mode: inserting new text
- Visual mode: text selection, copy (yank) & paste, operator on blocks
- Command-line mode: input command
- Full introduction: https://guide.freecodecamp.org/vim/modes/

Concept 2: Text Objects & Motions, manipulate text naturally (Influence many other apps)

- Command Structure: <command><number><text object or motion>.
- Number: how many times to repeat the command.
- Command: an operation, d(elete), c(hange), y(ank)
- Text object: w(ord), s(entence), p(aragraph), iw(inner word), aw(a word), is, as, ip, ap, i), a).
- Example:
  - daw: delete a word
  - c2w: change 2 words
  - ya): yank a round brackets

What Makes Vim Special: Three Concepts

Concept 1: Mode, do different things in different modes

- Normal mode: navigation and manipulation of text.
- Insert mode: inserting new text
- Visual mode: text selection, copy (yank) & paste, operator on blocks
- Command-line mode: input command
- Full introduction: https://guide.freecodecamp.org/vim/modes/

Concept 2: Text Objects & Motions, manipulate text naturally (Influence many other apps)

- Command Structure: <command><number><text object or motion>.
- Number: how many times to repeat the command.
- Command: an operation, d(elete), c(hange), y(ank)
- Text object: w(ord), s(entence), p(aragraph), iw(inner word), aw(a word), is, as, ip, ap, i), a).
- Example:
  - daw: delete a word
  - c2w: change 2 words
  - ya): yank a round brackets
- Full introduction: http://springest.io/vim-motions-and-command-language

What Makes Vim Special: Three Concepts

What Makes Vim Special: Three Concepts

Concept 3: Plugin, add extra power

What Makes Vim Special: Three Concepts

Concept 3: Plugin, add extra power

- Syntax highlighting (different languages)

What Makes Vim Special: Three Concepts

Concept 3: Plugin, add extra power

- Syntax highlighting (different languages)
- Auto-completion (different languages)

What Makes Vim Special: Three Concepts

Concept 3: Plugin, add extra power

- Syntax highlighting (different languages)
- Auto-completion (different languages)
- Linting (code analysis for different languages)

What Makes Vim Special: Three Concepts

Concept 3: Plugin, add extra power

- Syntax highlighting (different languages)
- Auto-completion (different languages)
- Linting (code analysis for different languages)
- Different color styles

What Makes Vim Special: Three Concepts

Concept 3: Plugin, add extra power

- Syntax highlighting (different languages)
- Auto-completion (different languages)
- Linting (code analysis for different languages)
- Different color styles
- Git integration

What Makes Vim Special: Three Concepts

Concept 3: Plugin, add extra power

- Syntax highlighting (different languages)
- Auto-completion (different languages)
- Linting (code analysis for different languages)
- Different color styles
- Git integration
- Fuzzy file finder / file tree

What Makes Vim Special: Three Concepts

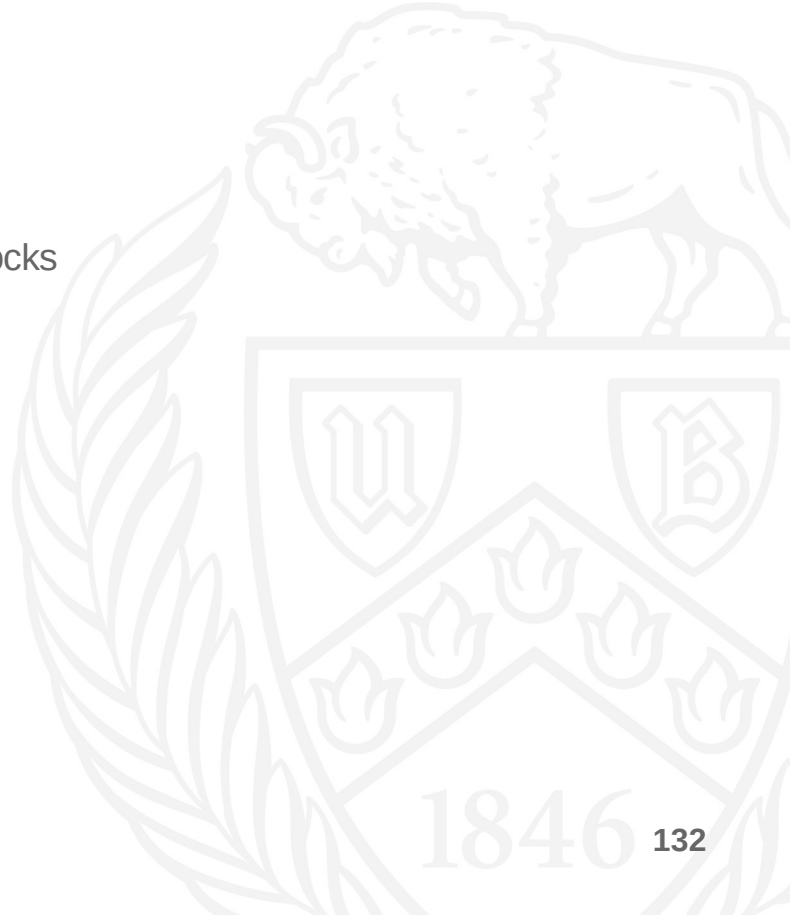
Concept 3: Plugin, add extra power

- Syntax highlighting (different languages)
- Auto-completion (different languages)
- Linting (code analysis for different languages)
- Different color styles
- Git integration
- Fuzzy file finder / file tree
- Multi-cursors, change multiple things at the same time

What Makes Vim Special: Three Concepts

Concept 3: Plugin, add extra power

- Syntax highlighting (different languages)
- Auto-completion (different languages)
- Linting (code analysis for different languages)
- Different color styles
- Git integration
- Fuzzy file finder / file tree
- Multi-cursors, change multiple things at the same time
- Visual mode: text selection, copy (yank) & paste, operator on blocks



What Makes Vim Special: Three Concepts

Concept 3: Plugin, add extra power

- Syntax highlighting (different languages)
- Auto-completion (different languages)
- Linting (code analysis for different languages)
- Different color styles
- Git integration
- Fuzzy file finder / file tree
- Multi-cursors, change multiple things at the same time
- Visual mode: text selection, copy (yank) & paste, operator on blocks
- Command-line mode: input command

What Makes Vim Special: Three Concepts

Concept 3: Plugin, add extra power

- Syntax highlighting (different languages)
- Auto-completion (different languages)
- Linting (code analysis for different languages)
- Different color styles
- Git integration
- Fuzzy file finder / file tree
- Multi-cursors, change multiple things at the same time
- Visual mode: text selection, copy (yank) & paste, operator on blocks
- Command-line mode: input command
- Full introduction: https://medium.com/@huntie/10-essential-vim-plugins-for-2018-39957190b7a9

What Makes Vim Special: Three Concepts

Concept 3: Plugin, add extra power

- Syntax highlighting (different languages)
- Auto-completion (different languages)
- Linting (code analysis for different languages)
- Different color styles
- Git integration
- Fuzzy file finder / file tree
- Multi-cursors, change multiple things at the same time
- Visual mode: text selection, copy (yank) & paste, operator on blocks
- Command-line mode: input command
- Full introduction: https://medium.com/@huntie/10-essential-vim-plugins-for-2018-39957190b7a9
- Many other tools: https://vimawesome.com/

What Makes Vim Special: Three Concepts

Concept 3: Plugin, add extra power

- Syntax highlighting (different languages)
- Auto-completion (different languages)
- Linting (code analysis for different languages)
- Different color styles
- Git integration
- Fuzzy file finder / file tree
- Multi-cursors, change multiple things at the same time
- Visual mode: text selection, copy (yank) & paste, operator on blocks
- Command-line mode: input command
- Full introduction: https://medium.com/@huntie/10-essential-vim-plugins-for-2018-39957190b7a9
- Many other tools: https://vimawesome.com/
- Package manager: https://github.com/junegunn/vim-plug

What Makes Vim Special: Three Concepts

Concept 3: Plugin, add extra power

- Syntax highlighting (different languages)
- Auto-completion (different languages)
- Linting (code analysis for different languages)
- Different color styles
- Git integration
- Fuzzy file finder / file tree
- Multi-cursors, change multiple things at the same time
- Visual mode: text selection, copy (yank) & paste, operator on blocks
- Command-line mode: input command
- Full introduction: https://medium.com/@huntie/10-essential-vim-plugins-for-2018-39957190b7a9
- Many other tools: https://vimawesome.com/
- Package manager: https://github.com/junegunn/vim-plug

How to Learn So Many Things?

What Makes Vim Special: Three Concepts

Concept 3: Plugin, add extra power

- Syntax highlighting (different languages)
- Auto-completion (different languages)
- Linting (code analysis for different languages)
- Different color styles
- Git integration
- Fuzzy file finder / file tree
- Multi-cursors, change multiple things at the same time
- Visual mode: text selection, copy (yank) & paste, operator on blocks
- Command-line mode: input command
- Full introduction: https://medium.com/@huntie/10-essential-vim-plugins-for-2018-39957190b7a9
- Many other tools: https://vimawesome.com/
- Package manager: https://github.com/junegunn/vim-plug

How to Learn So Many Things?

Use it.

What Makes Vim Special: Three Concepts

Concept 3: Plugin, add extra power

- Syntax highlighting (different languages)
- Auto-completion (different languages)
- Linting (code analysis for different languages)
- Different color styles
- Git integration
- Fuzzy file finder / file tree
- Multi-cursors, change multiple things at the same time
- Visual mode: text selection, copy (yank) & paste, operator on blocks
- Command-line mode: input command
- Full introduction: https://medium.com/@huntie/10-essential-vim-plugins-for-2018-39957190b7a9
- Many other tools: https://vimawesome.com/
- Package manager: https://github.com/junegunn/vim-plug

How to Learn So Many Things?

Use it.
Don’t push too much. Learn several things a time.

What Makes Vim Special: Three Concepts

Concept 3: Plugin, add extra power

- Syntax highlighting (different languages)
- Auto-completion (different languages)
- Linting (code analysis for different languages)
- Different color styles
- Git integration
- Fuzzy file finder / file tree
- Multi-cursors, change multiple things at the same time
- Visual mode: text selection, copy (yank) & paste, operator on blocks
- Command-line mode: input command
- Full introduction: https://medium.com/@huntie/10-essential-vim-plugins-for-2018-39957190b7a9
- Many other tools: https://vimawesome.com/
- Package manager: https://github.com/junegunn/vim-plug

How to Learn So Many Things?

Use it.
Don’t push too much. Learn several things a time.
Enjoy the process…

What Makes Vim Special: Three Concepts

Concept 3: Plugin, add extra power

- Syntax highlighting (different languages)
- Auto-completion (different languages)
- Linting (code analysis for different languages)
- Different color styles
- Git integration
- Fuzzy file finder / file tree
- Multi-cursors, change multiple things at the same time
- Visual mode: text selection, copy (yank) & paste, operator on blocks
- Command-line mode: input command
- Full introduction: https://medium.com/@huntie/10-essential-vim-plugins-for-2018-39957190b7a9
- Many other tools: https://vimawesome.com/
- Package manager: https://github.com/junegunn/vim-plug

How to Learn So Many Things?

Use it.
Don’t push too much. Learn several things a time.
Enjoy the process…
And, try the vim tutor app, $ vimtutor

How to Debug?

Debug Tools

- gdb: c++ debug app, https://darkdust.net/files/GDB%20Cheat%20Sheet.pdf
- jdb: java debug app, https://docs.oracle.com/javase/7/docs/technotes/tools/solaris/jdb.html
- pdb: python debug app, https://appletree.or.kr/quick_reference_cards/Python/Python%20Debugger%20Cheatsheet.pdf
- Many others ...

The end

The end

This is the start of a new journey

The end

This is the start of a new journey

Explore more!

The end

This is the start of a new journey

Explore more!

Try more!

The end

This is the start of a new journey

Explore more!

Try more!

Experiment more!

The end

This is the start of a new journey

Explore more!

Try more!

Experiment more!

And make more mistakes…

The end

This is the start of a new journey

Explore more!

Try more!

Experiment more!

And make more mistakes…

Most importantly

The end

This is the start of a new journey

Explore more!

Try more!

Experiment more!

And make more mistakes...

Most importantly

HAVE FUN!

THANK YOU!

THANK YOU!

There may be a followup email for a short survey for this course.