



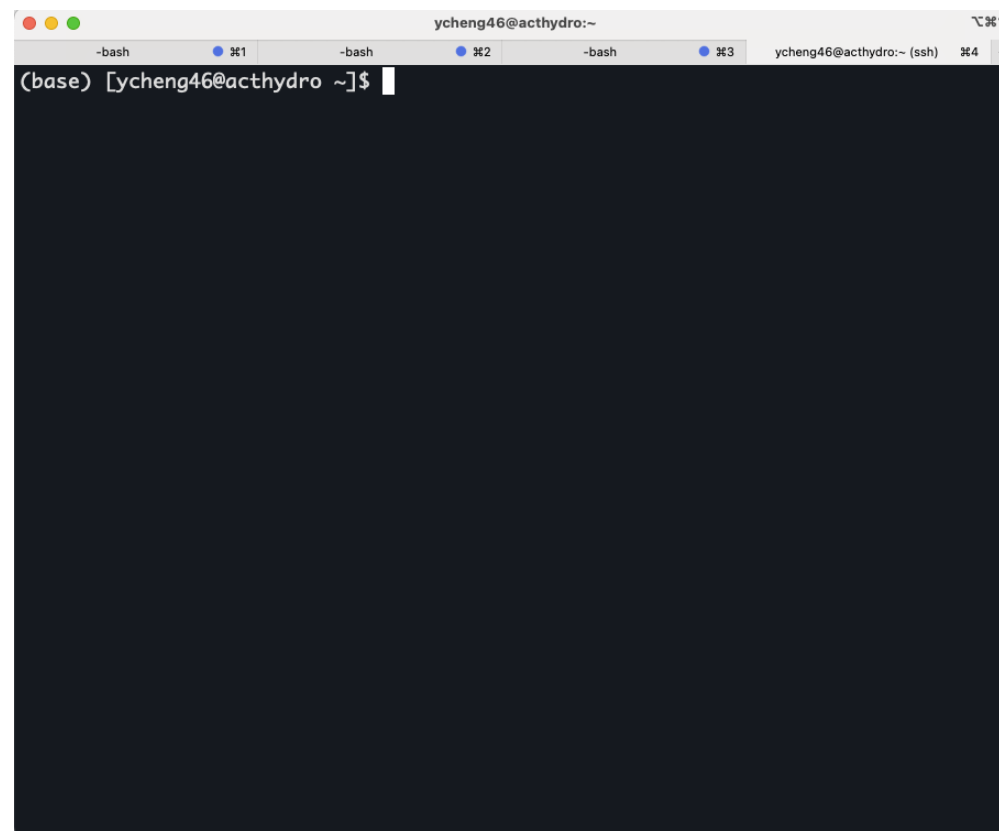
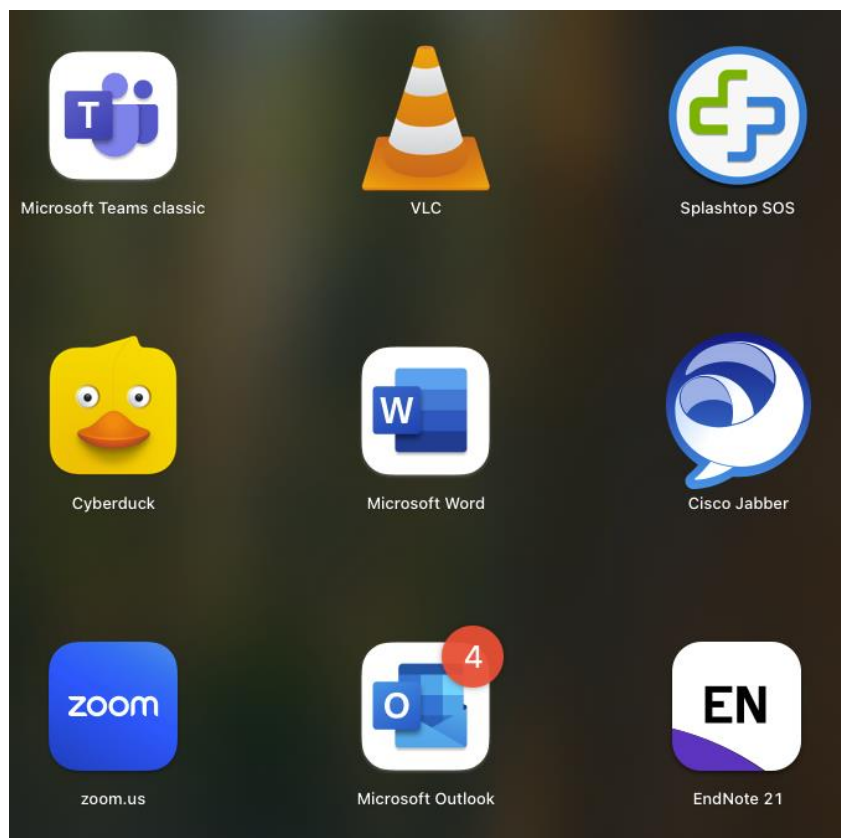
BASH SCRIPTS & COMMAND LINES

ERT 474/574

Open-Source Hydro Data Analytics

Oct 15th 2025

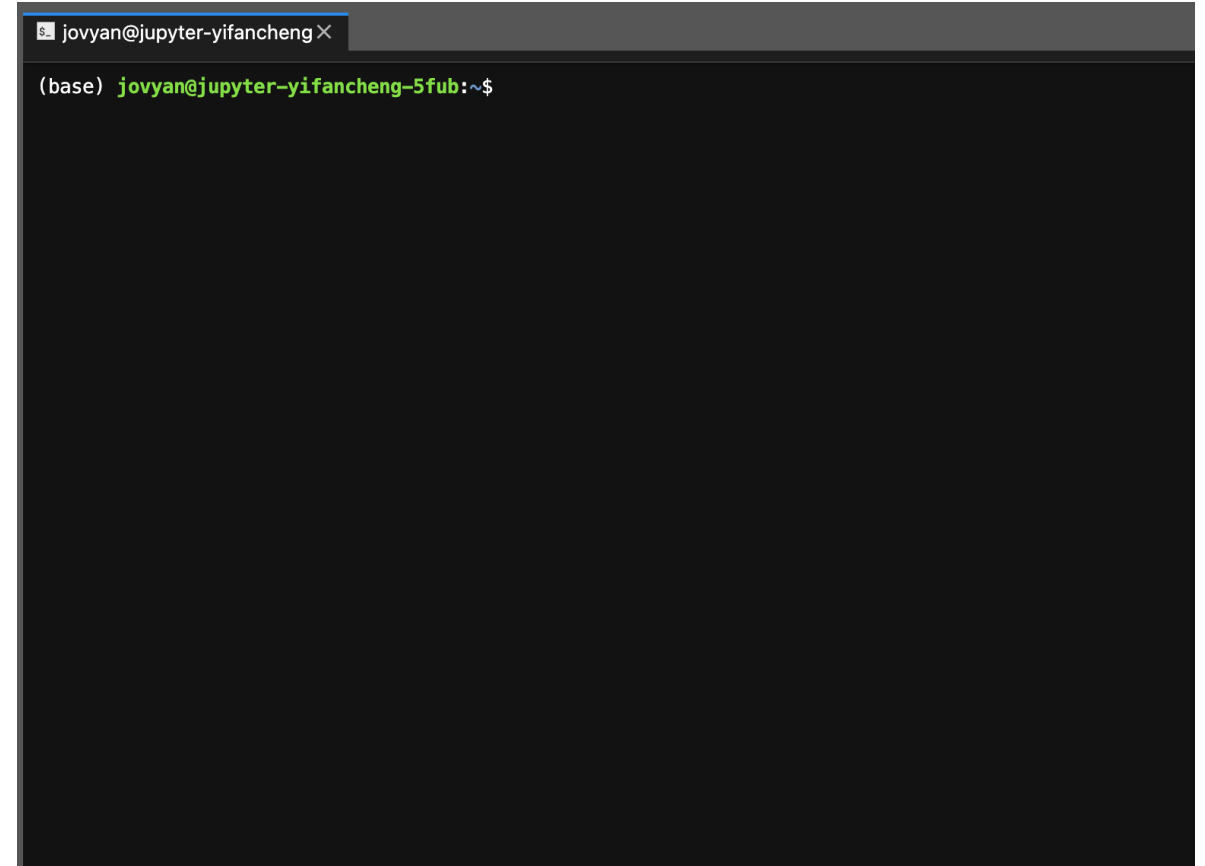
GUI versus Command Lines



What is the differences between the Graphical User Interface (GUI) and the Command Line?

What is bash script?

- A bash script is a file containing a sequence of commands that are executed by the bash program line by line. It allows you to perform a series of actions, such as navigating to a specific directory, creating a folder, and launching a process using the command line.

A terminal window with a dark background. The title bar at the top reads 'jovyan@jupyter-yifancheng'. The prompt line shows '(base) jovyan@jupyter-yifancheng-5fub:~\$'.

Why is it important to learn Bash?

Automation

Allow you to automate repetitive tasks and processes, saving time and reducing the risk of errors that can occur with manual execution.

Portability

Can be run on platforms and operating systems

- Unix
- Linux
- macOS
- Windows (emulators or virtual machines)

Flexibility

Highly customizable and can be easily modified to suit specific requirements.

Accessibility

Easy to write and don't require any special tools or software. They can be edited using any text editor, and most operating systems have a built-in shell interpreter.

Integration

Can be integrated with other tools & applications

- databases
- web servers
- cloud services

allowing for more complex automation & system management tasks.

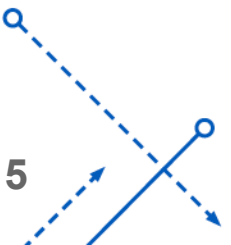
Open-Source Hydrologic Model

The screenshot shows the GitHub repository page for 'UW-Hydro / VIC'. The repository is public and has 144 issues, 4 pull requests, 47 watchers, 436 forks, and 288 stars. The 'Code' tab is selected, showing a list of files and their commit history. The files listed are .github, ci, docs, samples, tests, and tools. The commit history shows that the .github directory was fixed 9 years ago, ci was fixed 7 years ago, docs was updated 4 years ago, samples were speedup 8 years ago, tests were fixed 6 years ago, and tools were updated 10 years ago. The 'About' section on the right describes the repository as 'The Variable Infiltration Capacity (VIC) Macroscale Hydrologic Model' and provides links to the README, MIT license, Contributing guide, and Activity.

File	Commit Message	Time Ago
.github	fix location of github templates	9 years ago
ci	fixed datetime bug in unit tests	7 years ago
docs	try 2	4 years ago
samples	Speedup image driver initialization - in...	8 years ago
tests	fix travis tests again	6 years ago
tools	update helper script for running uncru...	10 years ago

In this class, we will learn how to configure and run the Variable Infiltration Capacity (VIC) hydrology model!

Here, you were provided the source code. What is your next step?

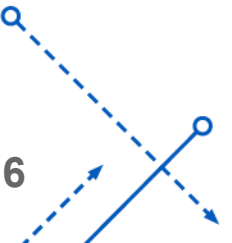


We need to use Command Lines to run VIC

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  JUPYTER

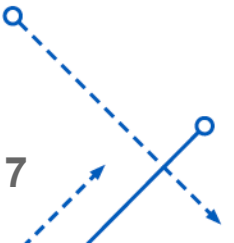
• @YifanCheng → /workspaces/VIC (master) $ ls
  LICENSE.txt  docs  mkdocs.yml  readme.md  samples  tests  tools  vic
○ @YifanCheng → /workspaces/VIC (master) $
```

Before we can run the models in GitHub Codespaces, we need to learn how to navigate the system.



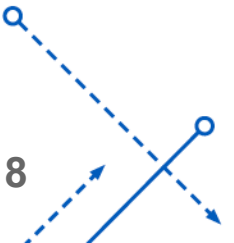
Commonly Used LINUX Utilities

- `cd` shell command for changing directory
- `head` displays the first n lines of data; if no “n” specified, it shows 10 lines
- `tail` display last n lines of file, if no “n”, the last 10 displayed
- `ls` lists files in the current directory
- `mkdir` creates a directory
- `more` allows you to scroll through a list of files one page at a time
- `mv` rename file; overwrite file; move to new location
- `pwd` print current working directory
- `rm` delete file
- `rmdir` delete directory
- `wc` counts lines, words, and characters



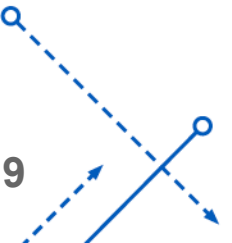
Text Searching

- `grep` useful utility for searching a file
- finds lines with the letter **q** in a file
 - `grep q filename`
- finds lines with the letter **q** and gives the line number in the file
 - `grep -n q filename`

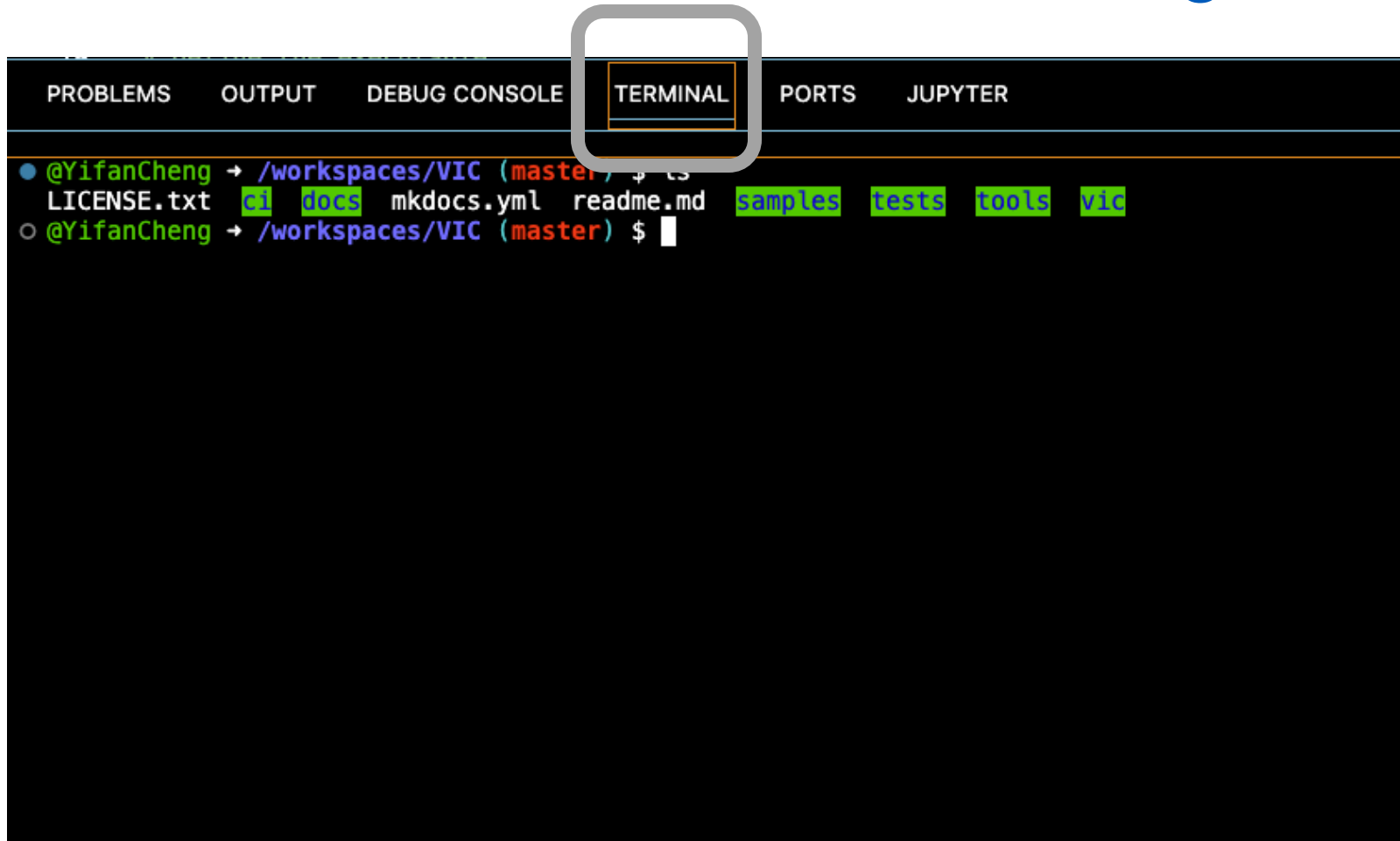


LINUX Utilities

- `cat` takes input from standard in or list of file(s) and displays them to standard output
- To use, type
 - `cat filename`
- Then press the return key.



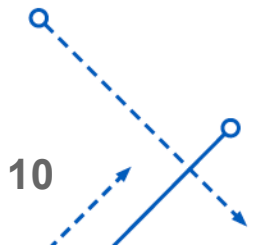
LINUX Fundamentals: Launching a Shell



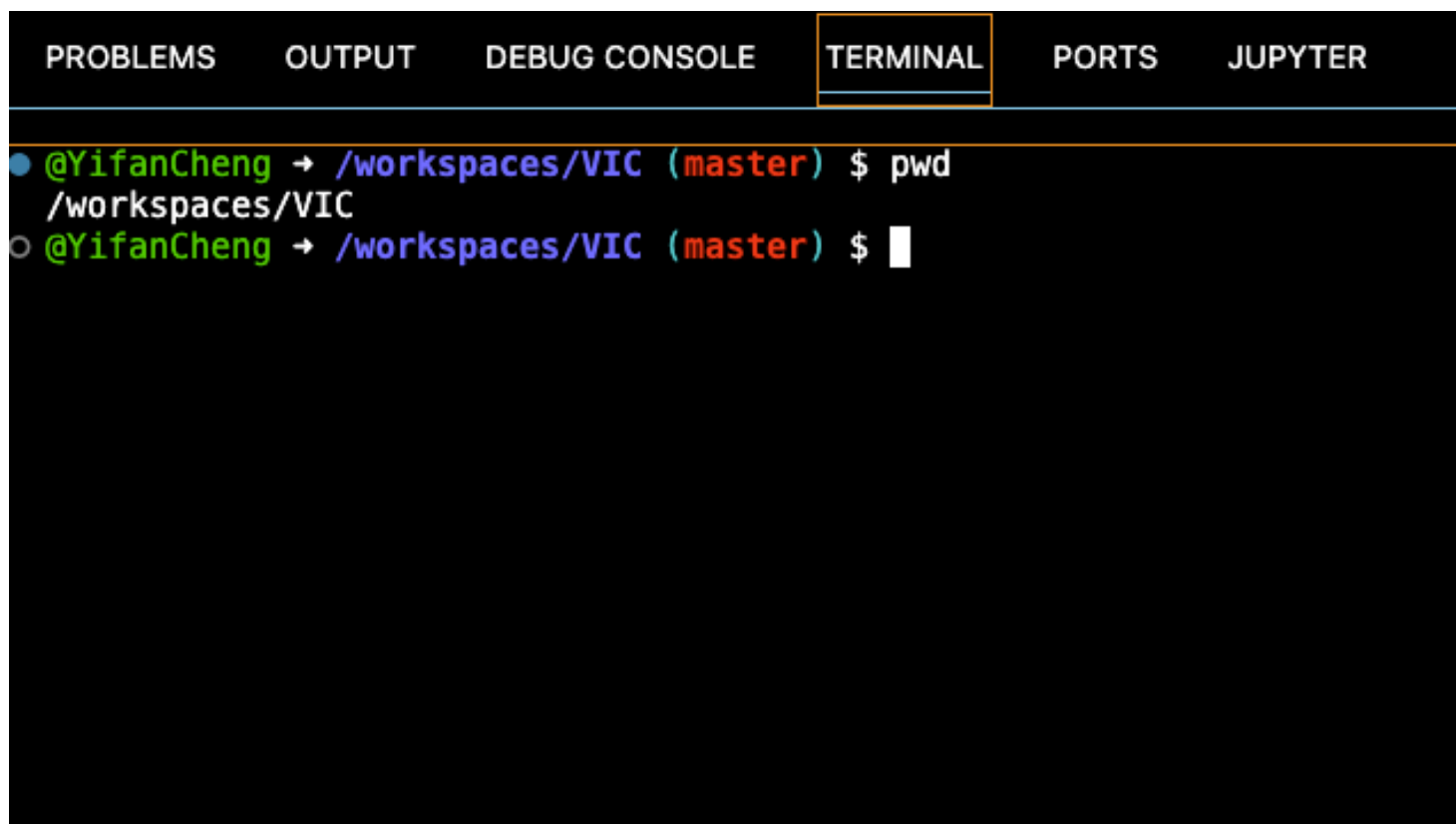
The screenshot shows the GitHub Codespace interface. At the top, there is a horizontal menu with several tabs: PROBLEMS, OUTPUT, DEBUG CONSOLE, **TERMINAL**, PORTS, and JUPYTER. The 'TERMINAL' tab is highlighted with a yellow border and a grey rounded rectangle. Below the tabs, the terminal window displays the following text:

```
● @YifanCheng → /workspaces/VIC (master) $ ls  
LICENSE.txt ci docs mkdocs.yml readme.md samples tests tools vic  
○ @YifanCheng → /workspaces/VIC (master) $
```

We only need to click “**TERMINAL**” in the GitHub Codespaces to launch a Shell.



LINUX Utilities: `pwd` example

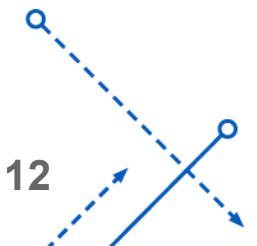


The screenshot shows a terminal window with a dark background. At the top, there is a navigation bar with tabs: PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL (which is highlighted with an orange border), PORTS, and JUPYTER. Below the tabs, the terminal shows two lines of text. The first line is a prompt: `@YifanCheng → /workspaces/VIC (master) $`, followed by the command `pwd`. The second line shows the output of the command: `/workspaces/VIC`. The third line shows the prompt again: `@YifanCheng → /workspaces/VIC (master) $` followed by a white cursor block.

To run `pwd`, for example, enter `pwd` on the command line and press the enter key. The result will be to print the current working directory to the terminal window.

LINUX Utilities: \$PATH

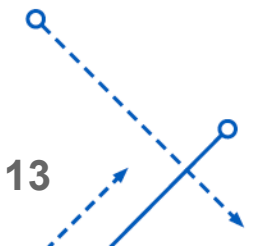
- The PATH provides:
- Locations where LINUX searches for commands.
- The search path is stored in the environment variable PATH
- To see what directories are in your path, type
 - `echo $PATH`
- You can add directories to your path, but the syntax will depend on the shell you are using.
- Note, you can also tell the computer to look in the current directory (.). For example:
 - `ls .` or `ls ./`



LINUX Utilities: Directory Structure

Recall that LINUX has a hierarchical directory structure

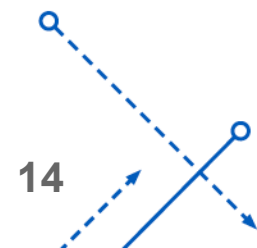
- You can access items via an absolute or relative path
- Absolute path is the full path to the directory or file beginning with /
- Relative path is the path to the directory or file relative to your current position
- For example, to change to the directory your current directory is located in using a relative path, enter
 - `cd ..`
- This will move you 'up' one directory.
- Alternatively, you could enter an absolute path (starting from /).
- For example, enter
 - `/the/path/to/the/directory`



LINUX Fundamentals: Locating programs

When you type `pwd` in the command line and hit enter, the shell searches directories for a program by the name of `pwd`

- For example, to find out where `pwd` is located, type
 - `which pwd`
- A common location for `pwd` is
 - `/bin` or `/usr/bin`



LINUX Fundamentals: Locating programs

- A common location for `pwd` is `/bin` or `/usr/bin`

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  JUPYTER
```

```
● @YifanCheng → /workspaces $ which pwd
  /usr/bin/pwd
○ @YifanCheng → /workspaces $
```

LINUX Fundamentals: Locating programs

- Listing the contents of the `/usr/bin` directory

```
@YifanCheng → /workspaces $ which pwd
/usr/bin/pwd
@YifanCheng → /workspaces $ cd /usr/bin/
@YifanCheng → /usr/bin $ ls
```

X11	ex	krb5-config	pdb3.13	svndump
'['	expand	krb5-config.mit	peekfd	svnserve
aclocal	expiry	ld	perl	svnsync
aclocal-1.17	expr	ld.bfd	perl5.40-x86_64-linux-gnu	svnversion
addr2line	f77	ld.so	perl5.40.1	sync
apropos	f95	ldd	perlbug	tabs
apt	factor	less	perldoc	tac
apt-cache	fallocate	lessecho	perlvp	tail
apt-cdrom	false	lessfile	perlthanks	tar
apt-config	fc-cache	lesskey	pg_config	taskset
apt-extracttemplates	fc-cat	lesspipe	pgrep	tbl
apt-ftparchive	fc-conflist	lexgrog	pic	tclsh
apt-get	fc-list	libnetcfg	pico	tclsh8.6
apt-mark	fc-match	libpng-config	piconv	tcltk-depends
apt-sortpkgs	fc-pattern	libpng16-config	pidof	tee
ar	fc-query	libtoolize	pidwait	tempfile
arch	fc-scan	libwmf-config	pinentry	test
as	fc-validate	link	pinentry-curses	tic
autoconf	fftw-wisdom	linux32	pinky	timeout
autoheader	fftw-wisdom-to-conf	linux64	pkg-config	tload
autom4te	fftwf-wisdom	ln	pkgconf	toe
automake	fftwl-wisdom	lnstat	pkgdata	top
automake-1.17	fftwq-wisdom	locale	kill	touch
autoreconf	fgrep	localedef	pl2pm	tput
autoscan	file	logger	pldd	tr
autoupdate	find	login	pmap	tree
awk	findmnt	logname	pod2html	troff
b2sum	flock	look	pod2man	true
base32	fmt	ls	pod2text	truncate
base64	fold	lsb_release	pod2usage	tset
basename	free	lsblk	podchecker	tsort

LINUX Fundamentals: Locating programs

Note: If the computer cannot find a program by the name you specified on the command line in either a pre-specified location or in the current directory, an error message will be printed to the terminal window.



LINUX Fundamentals: Locating programs

- For example, here the name for a program that does not exist (tomato) is entered. (Note you could also enter the name of a program that does exist, but if that directory is not in your path you will get the same error.)



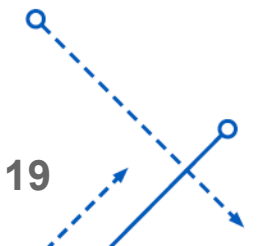
The screenshot shows a terminal window with tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL, PORTS, and JUPYTER. The TERMINAL tab is active. The prompt is `@YifanCheng → /usr/bin $`. The command `which tomato` has been entered. A red cross icon is visible to the left of the command line, indicating an error. A yellow circle highlights this red cross. A yellow box with the text "This red cross means an error message!" is overlaid on the terminal output area.

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS JUPYTER
@YifanCheng → /usr/bin $ which tomato
@YifanCheng → /usr/bin $
```

This red cross means an error message!

Commonly Used Metacharacters in the Shell

- **>** output redirection; writes standard output to a file
- **>>** output redirection; appends standard output to a file
- **<** input redirection; reads standard output from a file
- ***** File substitution wildcard; matches zero or more characters
- **|** Pipe symbol; sends output of one process to the input of another
- **&** Runs a command in the background



awk

- `awk` is another very useful program that you will use on the command line or in shell scripts:
 - `awk`
<https://www.gnu.org/software/gawk/manual/gawk.html>



Example Using `awk`

- Example using `awk` on the command line to extract the third column of data from a text file and redirect it to a new text file:

```
- awk '{print $3}' a1.txt > a13rdcol.txt
```



Text Editor

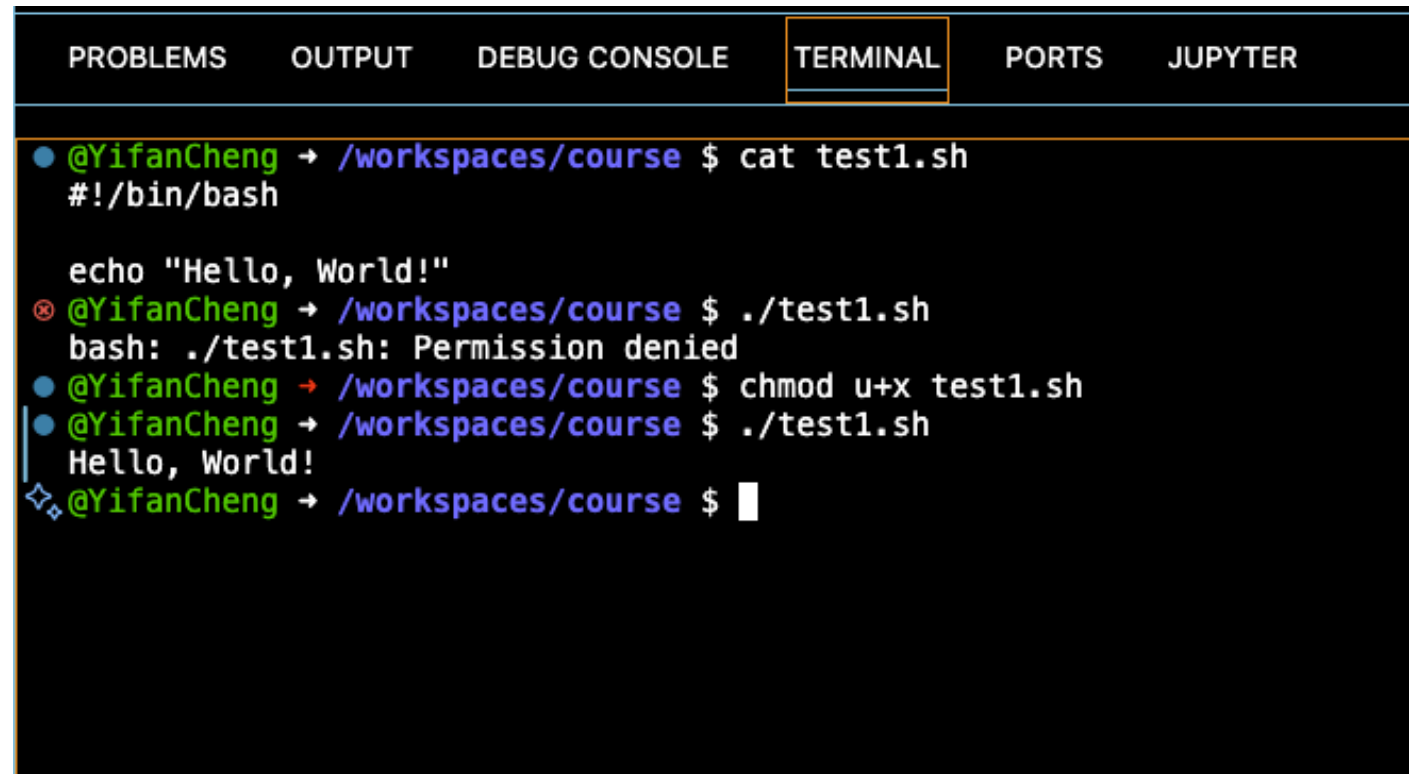
There are several text editors that commonly come with most UNIX and Linux systems. You will need to use one of these to be able to work on typical cluster systems. Example text editors include, but are not limited to:

- `vi`
- `vim`
- `emacs`
- `gedit`
- VS Code

Fortunately, in this class, you will not need to learn `vi` or `vim` because we have the VS Code GUI!

Example Writing and Running a Shell Script

- (1) Write the shell script.
- (2) Run the script. Note the current directory is specified by `./`
- (3) Update the permissions to executable with `chmod`
- (4) Run the script again, having reset permissions to executable



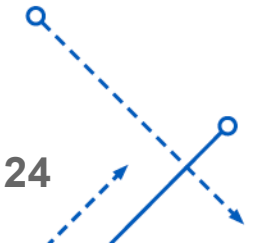
```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  JUPYTER

• @YifanCheng → /workspaces/course $ cat test1.sh
#!/bin/bash

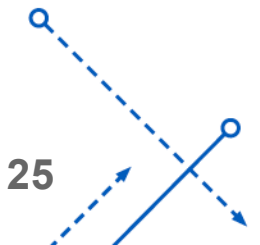
echo "Hello, World!"
⊗ @YifanCheng → /workspaces/course $ ./test1.sh
bash: ./test1.sh: Permission denied
• @YifanCheng → /workspaces/course $ chmod u+x test1.sh
• @YifanCheng → /workspaces/course $ ./test1.sh
Hello, World!
❖ @YifanCheng → /workspaces/course $
```

What's the best way to learn coding?

Through solving problems!



Case: prepare configuration files for all Ameriflux sites



Case: prepare configuration files for all Ameriflux sites

- When running hydrologic models, we usually use following command

```
## run job  
# -rmk pbs: use pbs for resource management  
# run on one processor  
$vic_exe -g $global_param
```

↓
Executable file

↓
Configuration file

Case: prepare configuration files
for all Ameriflux sites

What's inside a
configuration file?

Input data streams

Output spec

Timestep

Simulation period

&run_setup

```
static_file      = "./ameriflux_static_fields.C1152.US-SRG.nc"  
init_file       = "./ameriflux_init_fields.C1152.US-SRG.2009-12-31_23-00-00.nc"  
forcing_dir     = "forcing/US-SRG"
```

```
output_dir      = "./ufs_land_output/prod/US-SRG"  
output_frequency_s = 3600
```

```
timestep_seconds = 3600
```

```
! simulation_start is required  
! either set simulation_end or run_* or run_timesteps, priority  
!   1. simulation_end 2. run_[days/hours/minutes/seconds] 3. run_timesteps
```

```
simulation_start = "2010-01-01 00:00:00" ! start date [yyyy-mm-dd hh:mm:ss]
```

```
run_days        = 365 ! number of days to run  
run_hours       = 0  ! number of hours to run  
run_minutes     = 0  ! number of minutes to run  
run_seconds     = 0  ! number of seconds to run
```

```
run_timesteps   = 0  ! number of timesteps to run
```

```
location_start  = 1  
location_end    = 1
```

Naming conventions

```
static_file      = "./ameriflux_static_fields.C1152.US-SRG.nc"  
init_file       = "./ameriflux_init_fields.C1152.US-SRG.2009-12-31_23-00-00.nc"  
forcing_dir     = "forcing/US-SRG"
```

- **static_file** time-invariant fields, such as soil properties, vegetation types, etc.
- **init_file**. Initial condition file
- **forcing_dir** A directory contains time-variant meteorological forcing data, such as air temperature, precipitation, etc.

Naming conventions

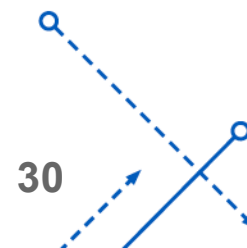
```
static_file = "ameriflux_static_fields.C1152.US-SRG.nc"
```

Prefix – information about file source
(ameriflux), file types (static file), etc.

Data specifications

- Spatial resolution
- Versions

Locations



Naming conventions

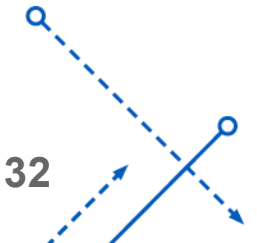
```
init_file = "../ameriflux_init_fields.C1152.US-SRG.2009-12-31_23-00-00.nc"
```

**What's the differences
in the name of initial
conditions files?**



Case: prepare configuration files for all Ameriflux sites

- Given a configuration file has already been created for US-SRG site, input data and file structures for all other sites are similar to the US-SRG site, please use a Bash script to create configuration files for all other sites.



Bash script cheat sheets

- General Bash Scripting Cheat Sheets
 - <https://devhints.io/bash>
- Vim Editor Cheat Sheet
 - <https://github.com/sk3pp3r/cheat-sheet-pdf/blob/master/pdf/vim-cheat-sheet.pdf>
- Single bracket or double bracket
 - <http://mywiki.woledge.org/BashFAQ/031>

