Shell Script & Useful Tools

Presenter: Yu-Cheng Chen

Date: 2023/8/9

Outline

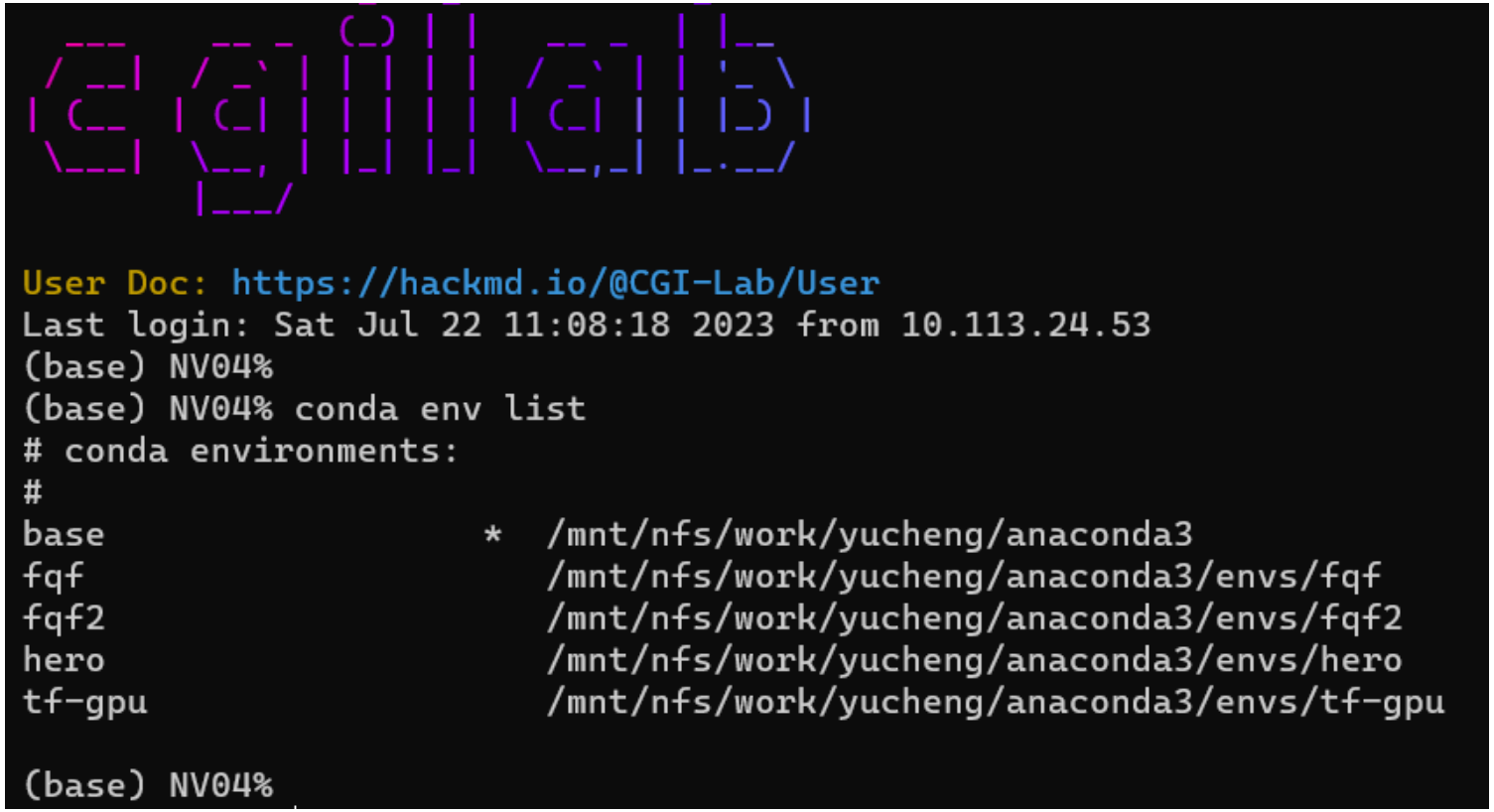
- Conda
- Dockerfile and Podman
- Shell Script
  - Variables
  - Mathematical Operations
  - Data Structures
  - Expressions
- Tmux

Virtual Environments

- Virtual environments in Python:
  - Configure a separate Python interpreter environment for each project to avoid dependency and version issues.
  - Avoid excessive installations of unnecessary packages to maintain a clean development environment and prevent contamination.
  - Easy to setup Pytorch and Tensorflow.
  - Easy to cleanup.

Virtual Environments

- Virtual environments in Python:
  1. conda
  2. pipenv
  3. virtualenv

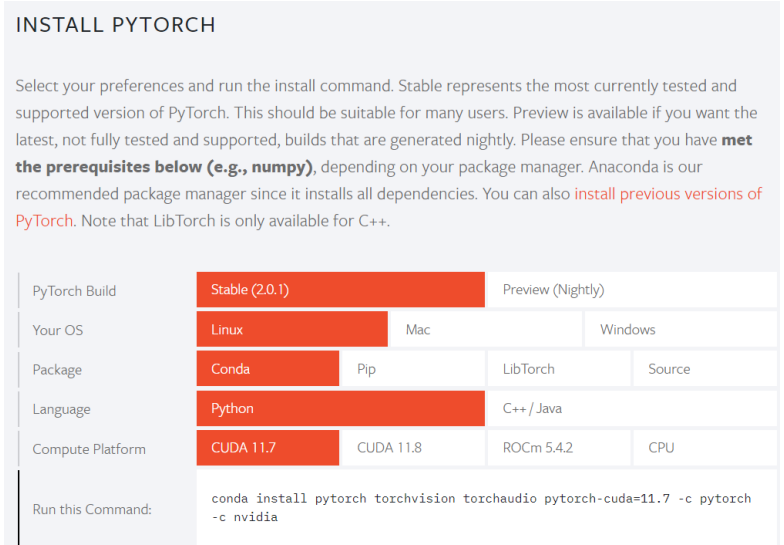


Q1: Conda

- Create conda environment for Tensorflow:
  - conda create -n tf-gpu tensorflow-gpu
  - conda activate tf-gpu
- Checkout Anaconda website:
  - https://docs.anaconda.com/anaconda/user-guide/tasks/tensorflow/

Q1: Conda

- Create conda environment for Pytorch:
  - conda create -n pytorch-gpu2 python=3.9
  - conda activate pytorch-gpu2
  - conda install pytorch torchvision torchaudio pytorch-cuda=11.7 -c pytorch -c nvidia
- Checkout Pytorch website:
  - https://pytorch.org/



Q1: Conda

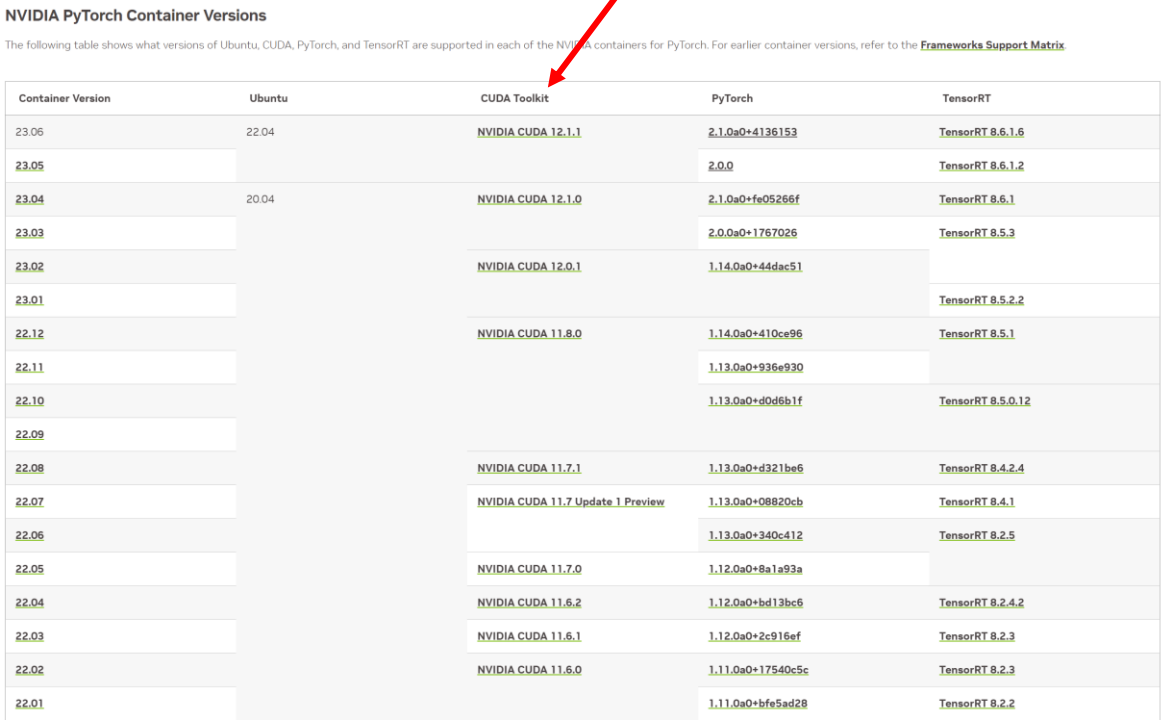
- Useful command:
  - conda env list
  - conda list
  - conda activate <name>
  - conda deactivate
  - conda env remove -n <name>

Q2: Dockerfile

- Create a container with access to GPUs.
  - nvidia-smi should work.
  - Install specific packages.

```
1 FROM nvcr.io/nvidia/pytorch:22.01-py3
2
3 RUN apt-get update \
4       && apt-get install -y tmux htop \
5       && rm -rf /var/lib/apt/lists/*
6
```

Have to match your driver's CUDA version.



NVIDIA PyTorch Container Versions

The following table shows what versions of Ubuntu, CUDA, PyTorch, and TensorRT are supported in each of the NVIDIA containers for PyTorch. For earlier container versions, refer to the Frameworks Support Matrix.

Container Version	Ubuntu	CUDA Toolkit	PyTorch	TensorRT
23.06	22.04	NVIDIA CUDA 12.1.1	2.1.0a0+4136153	TensorRT 8.6.1.6
23.05			2.0.0	TensorRT 8.6.1.2
23.04	20.04	NVIDIA CUDA 12.1.0	2.1.0a0+fe05266f	TensorRT 8.6.1
23.03			2.0.0a0+1767026	TensorRT 8.5.3
23.02		NVIDIA CUDA 12.0.1	1.14.0a0+44dac51	
23.01				TensorRT 8.5.2.2
22.12		NVIDIA CUDA 11.8.0	1.14.0a0+410ce96	TensorRT 8.5.1
22.11			1.13.0a0+936e930	
22.10			1.13.0a0+d0d6b1f	TensorRT 8.5.0.12
22.09				
22.08		NVIDIA CUDA 11.7.1	1.13.0a0+d321be6	TensorRT 8.4.2.4
22.07		NVIDIA CUDA 11.7 Update 1 Preview	1.13.0a0+08820cb	TensorRT 8.4.1
22.06			1.13.0a0+340c412	TensorRT 8.2.5
22.05		NVIDIA CUDA 11.7.0	1.12.0a0+8a1a93a	
22.04		NVIDIA CUDA 11.6.2	1.12.0a0+bd13bc6	TensorRT 8.2.4.2
22.03		NVIDIA CUDA 11.6.1	1.12.0a0+2c916ef	TensorRT 8.2.3
22.02		NVIDIA CUDA 11.6.0	1.11.0a0+17540c5c	TensorRT 8.2.3
22.01			1.11.0a0+bfe5ad28	TensorRT 8.2.2

https://docs.nvidia.com/deeplearning/frameworks/pytorch-release-notes/rel-23-06.html

Q2: Dockerfile and Podman

- podman build -t <image_name> -f Dockerfile .
- podman image ls

```
(base) NV04% podman image ls
REPOSITORY                  TAG         IMAGE ID      CREATED        SIZE
localhost/deepracer         2023        92ef05185647  3 weeks ago    20.7 GB
localhost/my_doc            latest      1ce633751886  3 weeks ago    15 GB
```

- podman run -it -d -v $PWD:/workspace --name=<container_name> <image_name> /bin/bash
- podman ps -a

```
(base) NV04% podman run -it -d -v $PWD:/workspace --name=MyDoc my_doc /bin/bash
0b7015ebf95b800179efd89a61a8deef4c29d077678a8288ec863e2dc2450523
(base) NV04% podman ps -a
CONTAINER ID  IMAGE                                  COMMAND               CREATED        STATUS                    PORTS                                           NAMES
7787c3aa05f6  quay.io/nordeus/heroic-rl-server:latest  ./HeroicRL --bind...  4 months ago   Exited (135) 4 months ago  0.0.0.0:8081->8081/tcp                          server
636265f34da1  localhost/my_doc:latest                                       3 weeks ago    Up 2 minutes ago                                                          test_dock
d605d0548836  k8s.gcr.io/pause:3.5                                         4 days ago     Exited (135) 47 hours ago  0.0.0.0:5050->5050/tcp, 0.0.0.0:6060->6060/tcp  05c9bf7ba396-infra
12ba17575eef  localhost/deepracer:2023               /bin/bash             4 days ago     Created                   0.0.0.0:5050->5050/tcp, 0.0.0.0:6060->6060/tcp  deepracer_train_0
0b7015ebf95b  localhost/my_doc:latest                /bin/bash             6 seconds ago  Up 6 seconds ago                                                          MyDoc
```

Q2: Dockerfile and Podman

- podman attach <container_name>

```
(base) NV04% podman attach MyDoc
root@0b7015ebf95b:/workspace# ls
@eaDir                              PR_Final_best.zip       cudnn-10.0-linux-x64-v7.6.5.32.tgz  latest_alpha_zero                          script_practice
Anaconda3-2022.05-Linux-x86_64.sh   RLFinal_backup          data                                libtorch                                   simple-shell-exercise
Atari_DQN_Framework                 SmartFoxServer_2X       deepracer                           libtorch-shared-with-deps-1.13.0+cu116.zip test
DLP                                 TCG                     final_project                       poetry-installer-error-ihrc4p1t.log        test_heroic
DLP_DQN                             anaconda3               fqf-iqn-qrdqn.pytorch               poetry-installer-error-nplu2hs6.log        ubuntu-18.04.6-desktop-amd64.iso
Dockerfile                          cplusplus_template.cpp  heroic-rl                           pommerman_experiment                       weights.bin
PPO_framework                       cuda                    hollow_nogo_alpha_zero              ray_results
```

- Detach: ctrl-p, ctrl-q (先按ctrl-p, 再按ctrl-q) (Use `exit` will shutdown the container)

- podman exec -it MyDoc <cmd>

```
(base) NV04% podman exec -it MyDoc ./script_practice/gpu_info.sh
Driver Version: 510.85.02
CUDA Version: 11.6
GPUs: 4
1: NVIDIA GeForce GTX 1080 Ti
2: NVIDIA GeForce GTX 1080 Ti
3: NVIDIA GeForce GTX 1080 Ti
4: NVIDIA GeForce GTX 1080 Ti
```

Shell Script - Variables

- Variable_name=value. No space in both sides.
- Bash does not have the concept of datatypes, but you can consider the default type as string.
- Use $ or ${ } to get the value of variable.
- Example:

```
1 #!/bin/bash
2
3 str1="My score is "
4 math=90
5 english=87.5
6
7 echo $math+$english
8 echo $str1+$english
9 echo $str1$english
```

```
(base) NV10% ./test.sh
90+87.5
My score is +87.5
My score is 87.5
```

Shell Script - Variables

- String append:

```
1 #!/bin/bash
2
3 str1="My score is "
4 math=90
5 english=87.5
6 str1=$str1$math
7 echo $str1
```

```
(base) NV10% ./test.sh
My score is 90
```

Shell Script - Variables

- String split:

```
 1 #!/bin/bash
 2
 3 str1="My score is "
 4 math=90
 5 english=87.5
 6 str1=$str1$math
 7 echo $str1
 8
 9 for k in $str1
10 do
11   echo $k
12 done
13
```

Use for loop (will be covered later)

```
(base) NV10% ./test.sh
My score is 90
My
score
is
90
```

Shell Script - Math Operations

```
1 #!/bin/bash
2
3 n=2
4 m=3
5
6 echo $n+$m
```

```
(base) NV10% ./test.sh
2+3
```

- Can not use + directly.
- Can use $(( )) in integer calculation.
  - +, -, *, /, %, ** work.
  - Range is long long.

```
1 #!/bin/bash
2
3 n=2
4 m=63
5
6 echo $((n**m-1))
```

```
(base) NV10% ./test.sh
9223372036854775807
```

- For complex math operations, use bc (basic calculator).
  - +, -, *, /, %, ^, sqrt… work.
  - echo “expression” | bc

Shell Script - Math Operations

- Examples of using bc:
  - Big number calculation.

```
1 #!/bin/bash
2
3 n=10
4 m=100
5
6 echo "sqrt($n^$m)" | bc
```

```
(base) NV10% ./test.sh
10000000000000000000000000000000000000000000000000
```

Shell Script - Math Operations

- Examples of using bc:
  - Float number calculation.

```
1 #!/bin/bash
2
3 n=17
4 m=7
5
6 echo "17/7" | bc
```

```
(base) NV10% ./test.sh
2
```

```
1 #!/bin/bash
2
3 n=17
4 m=7
5
6 echo "17/7" | bc -l
```

```
(base) NV10% ./test.sh
2.42857142857142857142
```

```
1 #!/bin/bash
2
3 n=17
4 m=7
5
6 echo "scale=3; 17/7" | bc
```

```
(base) NV10% ./test.sh
2.428
```

Shell Script - Math Operations

- Examples of using bc:
  - Binary operation.

```
1 #!/bin/bash
2
3 n=11000000
4
5 echo "obase=10; ibase=2; $n" | bc
```

```
(base) NV10% ./test.sh
192
```

Shell Script - Math Operations

- Examples of using bc:
  - Store the result of complete operation.

```
1 #!/bin/bash
2
3 n=1
4
5 e=`echo "scale=5; e($n)" | bc -l`
6 echo "exponential e = $e"
```

```
(base) NV10% ./test.sh
exponential e = 2.71828
```

- -l: standard math library.
- e: a built-in math functions.
- Use ``` or $( ) to store the result of an output.

Shell Script - Data Structures

- Bash Array can store a large group of variables.
- There are two types of bash arrays:
  - Indexed: the array is referred via integers or numbers.
  - Associative: the array is referred via strings or a set of characters and words.
- Bash does not support multidimensional arrays.

Shell Script - Data Structures

- Indexed array:
  - Declare:

```
declare -a Indexed
```

  - Assign values:
    - Method 1:

```
Indexed[0]=car
Indexed[1]=plane
Indexed[2]=bike
```

```
Indexed[0]=car
Indexed[2]=bike
```

    - Method 2:

```
Indexed=(car plane bike)
```

Shell Script - Data Structures

- Associative array:
  - Declare:

```
declare -A Associative
```

  - Assign values:
    - Method 1:

```
Associative[color]=blue
Associative[type]=car
Associative[topspeed]=200
```

    - Method 2:

```
declare -A Associative=( [color]=blue [type]=car [topspeed]=200 )
```

Shell Script - Data Structures

- Reference and print an array element:
  - Use ${name[index]}
  - Indexed array:

```
Indexed[0]=car
Indexed[1]=plane
Indexed[2]=bike

echo ${Indexed[1]}
```

  - Associative array:

```
Associative[color]=blue
Associative[type]=car
Associative[topspeed]=200

echo ${Associative[type]}
```

Shell Script - Data Structures

- Add a variable to a Bash Array:
  - Use += operator.
  - Indexed array:

```
Indexed[0]=car
Indexed[1]=plane
Indexed[2]=bike
Indexed+=(motorcycle)
```

  - Associative array:

```
Associative[color]=blue
Associative[type]=car
Associative[topspeed]=200
Associative+=([engine]=gas)
```

Shell Script - Data Structures

- Remove elements in Bash Array:
  - Use unset.
  - Indexed array:

```
Indexed[0]=car
Indexed[1]=plane
Indexed[2]=bike
unset Indexed[1]
```

  - Associative array:

```
Associative[color]=blue
Associative[type]=car
Associative[topspeed]=200
unset Associative[topspeed]
```

Shell Script - Data Structures

- Loop through an array:

```
Associative[color]=blue
Associative[type]=car
Associative[topspeed]=200

for i in "${Associative[@]}"
do
  echo $i
done
```

```
(base) NV10% ./array.sh
blue
car
200
```

```
for key in "${!Associative[@]}"
do
  echo "$key": "${Associative[$key]}"
done
```

```
(base) NV10% ./array.sh
color: blue
type: car
topspeed: 200
```

Shell Script - Expressions

- for loop:

```
for var in something_iterable
do
    #something
done
```

```
#!/bin/bash

for i in `seq 1 10`
do
  echo $i
done
```

```
#!/bin/bash

for ((i=1; i<=10; i++))
do
  echo $i
done
```

Shell Script - Expressions

- while loop:

```
while [ condition ]
do
    #something
done
```

```bash
#!/bin/bash

n=10
i=1
while [ $i -le $n ]
do
  echo $i
  i=$(($i+1))
done
```

```bash
#!/bin/bash

n=10
i=1
while :
do
  echo $i
  i=$(($i+1))

  if [ $i -gt $n ]; then
    break
  fi
done
```

```bash
#!/bin/bash

while read line
do
  echo $line
done < test.txt
```

Shell Script - Expressions

- if:

```
if [ cond1 ]; then
elif [ cond2 ]; then
else
fi
```

```
#!/bin/bash

str1="N"

if [ $str1 == "y" ] || [ $str1 == "Y" ]; then
  echo "success"
else
  echo "error"
fi
```

- ==, != are for strings.
- -gt, -lt, -ge, -le, -eq, -ne are for numbers.
- Use &&, || for multiple conditions.

Shell Script - Expressions

- case:

```
case $var in
        "a"  )
                # something
    ;;
        "b"  )
                # something
    ;;
    *)
                #something
    ;;
esac
```

```
#!/bin/bash

str1=b

case $str1 in
  "a")
    echo "a"
  ;;
  "b")
    echo "b"
  ;;
  "c")
    echo "c"
  ;;
  *)
    echo "d"
  ;;
esac
```

Shell Script - Expressions

- awk
  - Alfred Aho, Peter Weinberger, and Brian Kernighan.
  - Processing text files.
  - Text analysis.
- awk structure
  - awk ‘pattern { action }’
  - pattern: BEGIN, END, regex.
  - action: if, while, for …… grammar similar to C.

Shell Script - Expressions

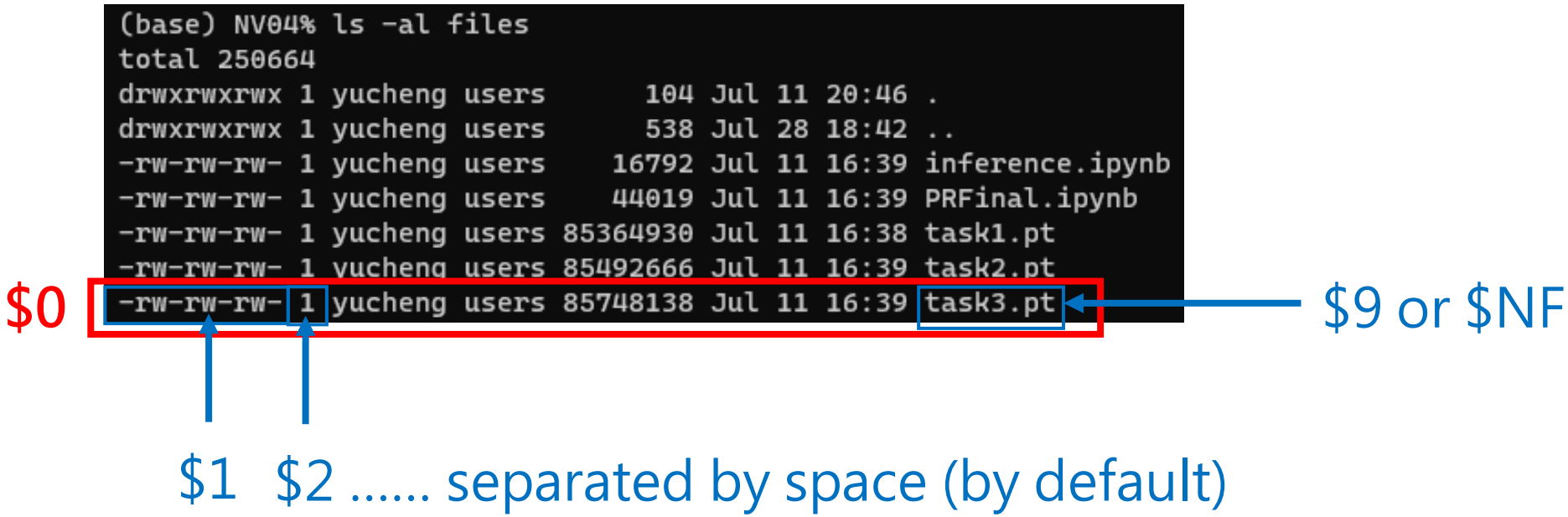
- awk example:

```
(base) NV04% ls -al files
total 250664
drwxrwxrwx 1 yucheng users      104 Jul 11 20:46 .
drwxrwxrwx 1 yucheng users      538 Jul 28 18:42 ..
-rw-rw-rw- 1 yucheng users    16792 Jul 11 16:39 inference.ipynb
-rw-rw-rw- 1 yucheng users    44019 Jul 11 16:39 PRFinal.ipynb
-rw-rw-rw- 1 yucheng users 85364930 Jul 11 16:38 task1.pt
-rw-rw-rw- 1 yucheng users 85492666 Jul 11 16:39 task2.pt
-rw-rw-rw- 1 yucheng users 85748138 Jul 11 16:39 task3.pt
```

```
(base) NV04% ls -al files | awk '{print $1}'
total
drwxrwxrwx
drwxrwxrwx
-rw-rw-rw-
-rw-rw-rw-
-rw-rw-rw-
-rw-rw-rw-
-rw-rw-rw-
```

Shell Script - Expressions

- awk example:



Shell Script - Expressions

- awk split example:
  - Separator: -F $'FS'$
  - Multiple separator: -F $'[FS]'$

```
(base) NV04% ls *.ckpt.index
6_Step-658.ckpt.index
(base) NV04% ls *.ckpt.index | awk -F '-' '{print $1, $2}'
6_Step 658.ckpt.index
(base) NV04% ls *.ckpt.index | awk -F '[-_.]' '{print $1, $3}'
6 658
```

Shell Script - Other Useful Tricks

- Read from stdin:

```
#!/bin/bash
read -p "Please enter the name of the file: " fileName
echo "The filename you entered is: $fileName"
```

read -p <prompt statement> <variable_name>

- Add `Help` command in your script:

```
if [ "$1" == "--help" ] || [ "$1" == "-h" ];
then
        echo   “Usage <arg1> <arg2> …”
        exit 0
fi
```

Q3 using awk

- Q3: Write a script that shows:

```
(base) NV27% ./gpu_info.sh
Driver Version: 520.61.05
CUDA Version: 11.8
GPUs: 4
1: NVIDIA GeForce RTX 3080 Ti
2: NVIDIA GeForce RTX 3080 Ti
3: NVIDIA GeForce RTX 3080 Ti
4: NVIDIA GeForce RTX 3080 Ti
```

- Ans:

```
#!/bin/bash

nvidia-smi -q | awk 'BEGIN {cnt=1}
  {
    if ($1=="Driver" && $2=="Version")
      printf "Driver Version: %s\n", $4
    else if ($1=="CUDA" && $2=="Version")
      printf "CUDA Version: %s\n", $4
    else if ($1=="Attached" && $2=="GPUs")
      printf "GPUs: %s\n", $4
    else if ($1=="Product" && $2=="Name"){
      printf "%d: %s %s %s %s %s\n", cnt, $4, $5, $6, $7, $8
      cnt+=1
    }
  }'
```

Q4 using awk

- Q4: Write a script that shows:

```
(base) NV04% ./cpu_info.sh
Architecture:                    x86_64
CPU(s):                          56
Model name:                      Intel(R) Xeon(R) CPU E5-2683 v3 @ 2.00GHz
```

- Ans:

```
#!/bin/bash

lscpu | awk '{
    if ($1=="Architecture:")
      print $0
    else if ($1=="CPU(s):")
      print $0
    else if ($1=="Model" && $2=="name:")
      print $0
  }'
```

Q6 using awk

- Q6: Write a script that shows:

```
(base) yucheng@yucheng-System-Product-Name:~$ ./get_weights_size.sh
=======================================
file name    size(KB)    size(MB)
=======================================
task1.pt    85364930    81.41
task2.pt    85492666    81.53
task3.pt    85748138    81.78
=======================================
total    256605734    244.72
```

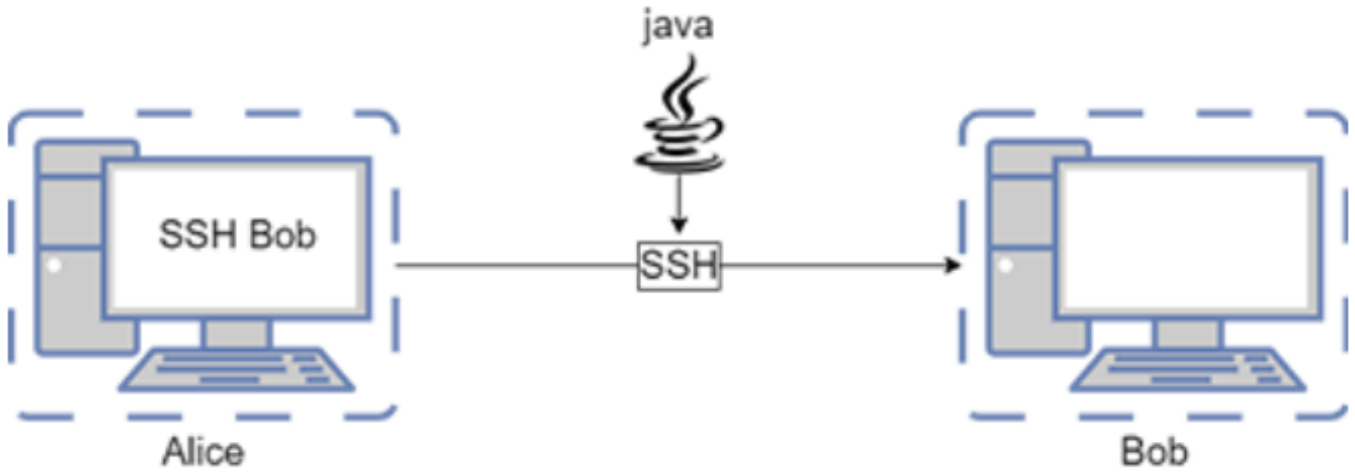
- Ans:

```
1 #!/bin/bash
2 echo "======================================="
3 echo "file name    size(KB)    size(MB)"
4 echo "======================================="
5
6 ls -al files | awk 'BEGIN { size=0 }
7   $NF~/.*.pt/ {size+=$5; printf "%s    %d    %.2f\n",$NF, $5, $5/(1024*1024)}
8   END {printf "=======================================\ntotal    %d    %.2f\n", size, size/(1024*1024)}
9   '
```

Using regular expressions to match lines.
Syntax: word ~ /regex/

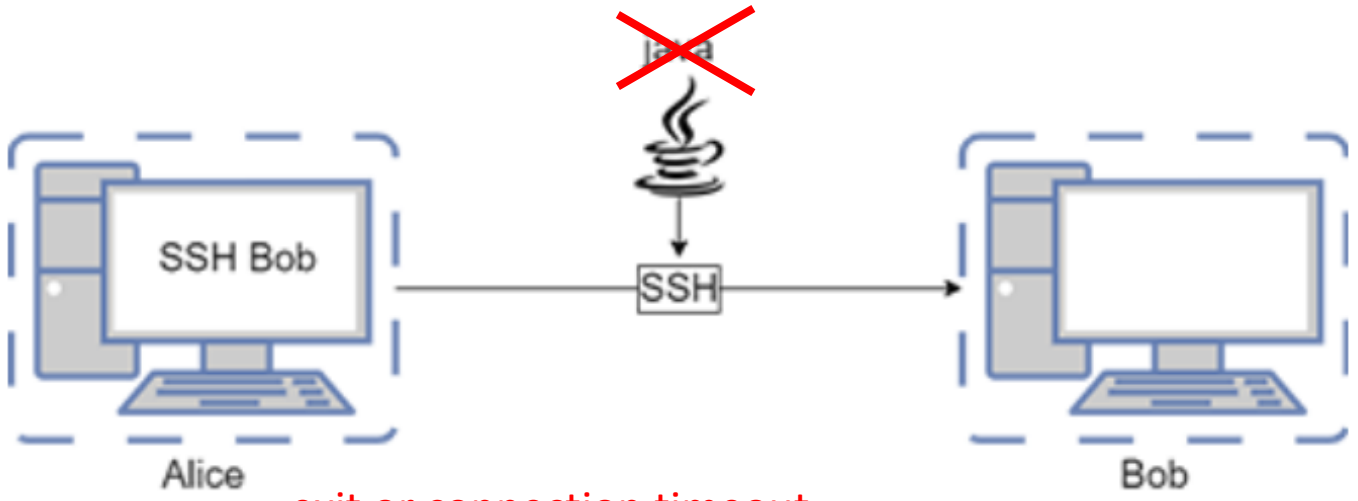
Tmux

- Terminal multiplexer
- Why tmux ?
  - Server kills your process when you disconnect !



Tmux

- Terminal multiplexer
- Why tmux ?
  - Server kills your process when you disconnect !

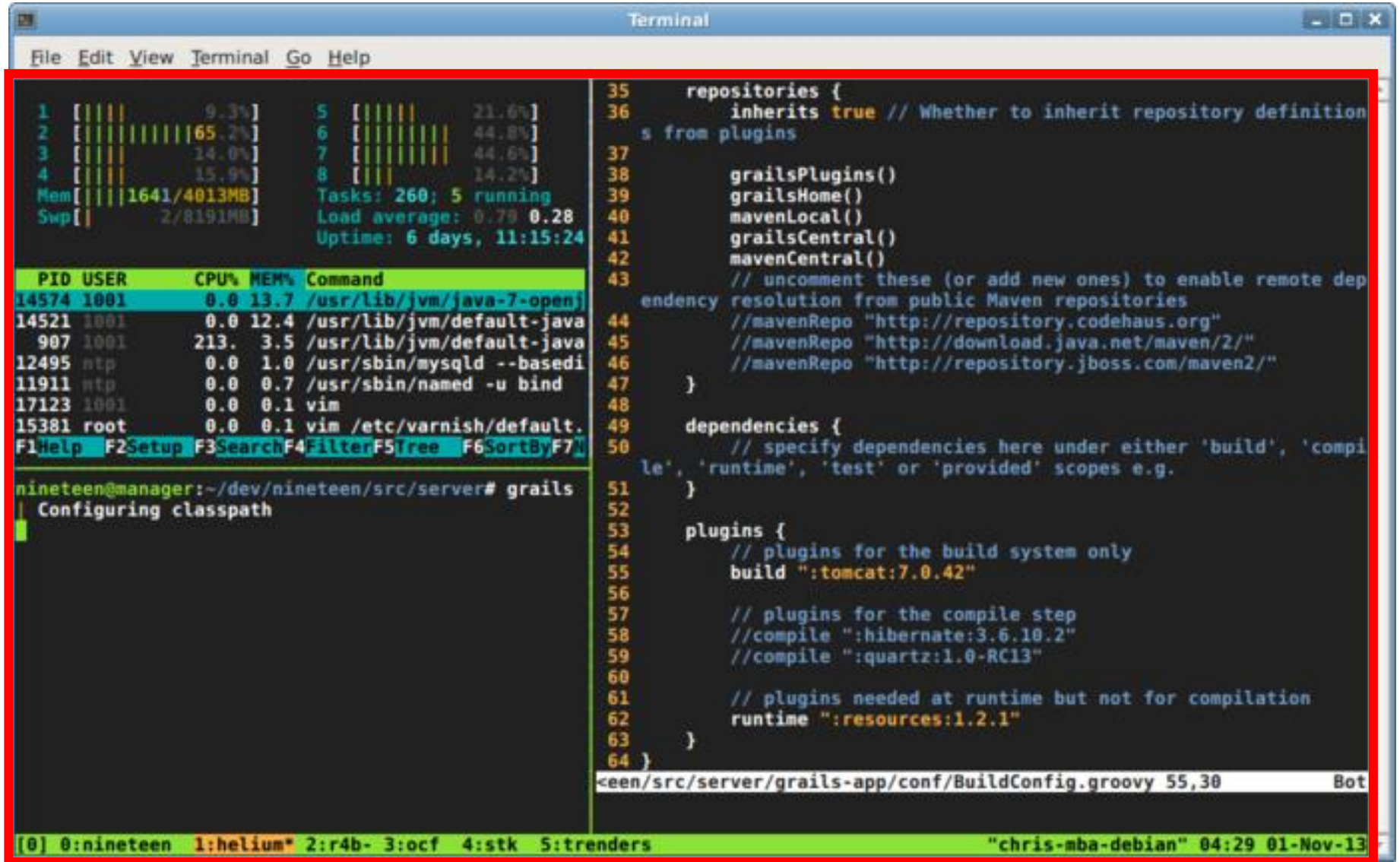


Tmux

- Alice ssh to Bob and run Tmux.
- The process that Alice run is running on the Tmux server, not related to ssh.
- The disconnection of ssh has no effect.

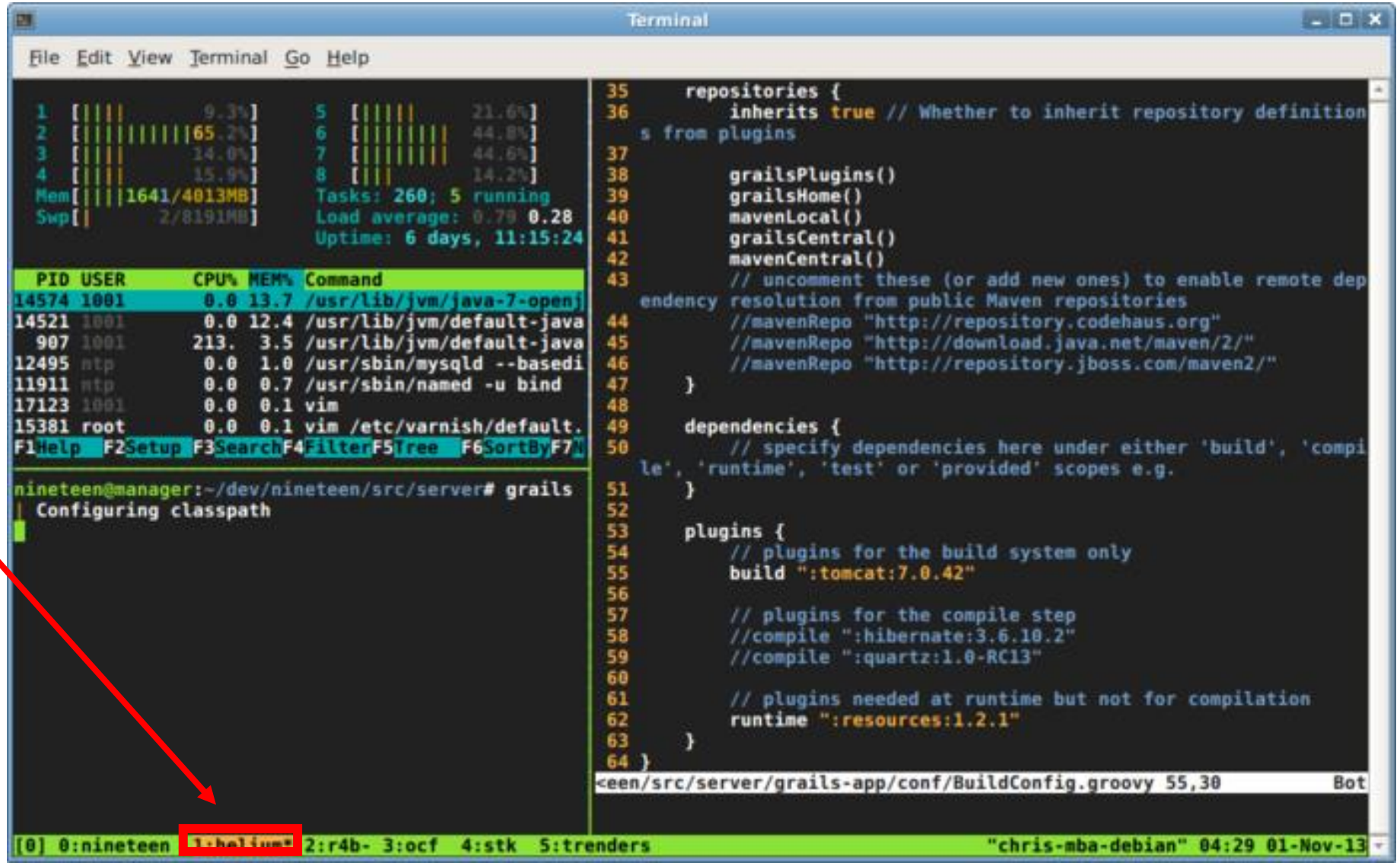
Tmux UI

session



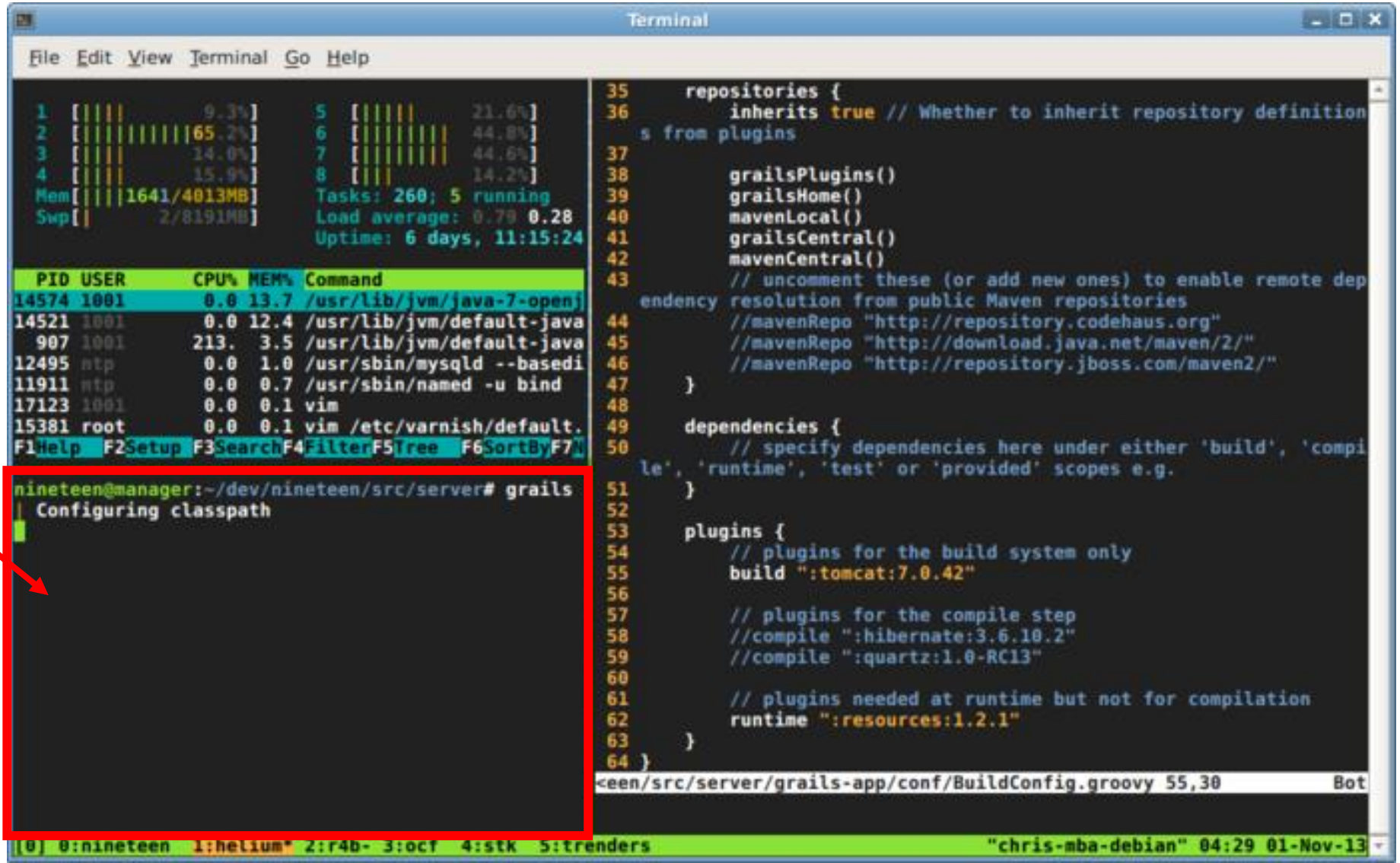
Tmux UI

window



Tmux UI

pane


Terminal
File Edit View Terminal Go Help
1 [|||| 9.3%]
2 [|||||||||||65.2%]
3 [||||| 14.0%]
4 [||||| 15.9%]
5 [|||||| 21.6%]
6 [||||||||| 44.8%]
7 [||||||||| 44.6%]
8 [|||| 14.2%]
Mem[||||1641/4013MB]
Swp[| 2/8191MB]
Tasks: 260; 5 running
Load average: 0.79 0.28
Uptime: 6 days, 11:15:24
PID USER CPU% MEM% Command
14574 1001 0.0 13.7 /usr/lib/jvm/java-7-openj
14521 1001 0.0 12.4 /usr/lib/jvm/default-java
907 1001 213. 3.5 /usr/lib/jvm/default-java
12495 ntp 0.0 1.0 /usr/sbin/mysqld --basedi
11911 ntp 0.0 0.7 /usr/sbin/named -u bind
17123 1001 0.0 0.1 vim
15381 root 0.0 0.1 vim /etc/varnish/default.
F1Help F2Setup F3SearchF4FilterF5Tree F6SortByF7
nineteen@manager:~/dev/nineteen/src/server# grails
| Configuring classpath
repositories {
inherits true // Whether to inherit repository definitions from plugins
grailsPlugins()
grailsHome()
mavenLocal()
grailsCentral()
mavenCentral()
// uncomment these (or add new ones) to enable remote dependency resolution from public Maven repositories
//mavenRepo "http://repository.codehaus.org"
//mavenRepo "http://download.java.net/maven/2/"
//mavenRepo "http://repository.jboss.com/maven2/"
}
dependencies {
// specify dependencies here under either 'build', 'compile', 'runtime', 'test' or 'provided' scopes e.g.
}
plugins {
// plugins for the build system only
build ":tomcat:7.0.42"
// plugins for the compile step
//compile ":hibernate:3.6.10.2"
//compile ":quartz:1.0-RC13"
// plugins needed at runtime but not for compilation
runtime ":resources:1.2.1"
}
}
<een/src/server/grails-app/conf/BuildConfig.groovy 55,30 Bot
[0] 0:nineteen 1:helium* 2:r4b- 3:ocf 4:stk 5:trenders
"chris-mba-debian" 04:29 01-Nov-13

Tmux Command

- a + b: press together (同時按)
- a , b: press a then press b (先按a, 再按b)

Action	Command
Split pane (horizontal)	Ctrl + b , %
Split pane (vertical)	Ctrl + b , “
Choose pane	Ctrl + b , →
Close pane	Ctrl + d
New window	Ctrl + b , c
Previous window	Ctrl + b , p
Next window	Ctrl + b , n

Tmux Command

Action	Command
Choose window	Ctrl + b , <number>
Resize pane	Ctrl + b + →
Rolling mode	Ctrl + b , PageUp
Exit Rolling mode	q
Detach	Ctrl + b , d

Tmux Command

Action	Command (outside tmux)
Install	apt install tmux
Enter tmux (new session)	tmux
Attach to tmux session	tmux attach –t <name>
list tmux session	tmux ls
Attach to latest session	tmux a
Kill session	tmux kill-session -t <name>

Q5 tmux script

```
#!/bin/bash

SESSION_NAME="exercise"
tmux new-session -d -s $SESSION_NAME
tmux split-window -h -t $SESSION_NAME
tmux select-pane -t $SESSION_NAME -L
tmux send-keys -t $SESSION_NAME 'ping 8.8.8.8' C-m
tmux select-pane -t $SESSION_NAME -R
tmux send-keys -t $SESSION_NAME 'watch -n 2 -d nvidia-smi' C-m
```

Tmuxinator

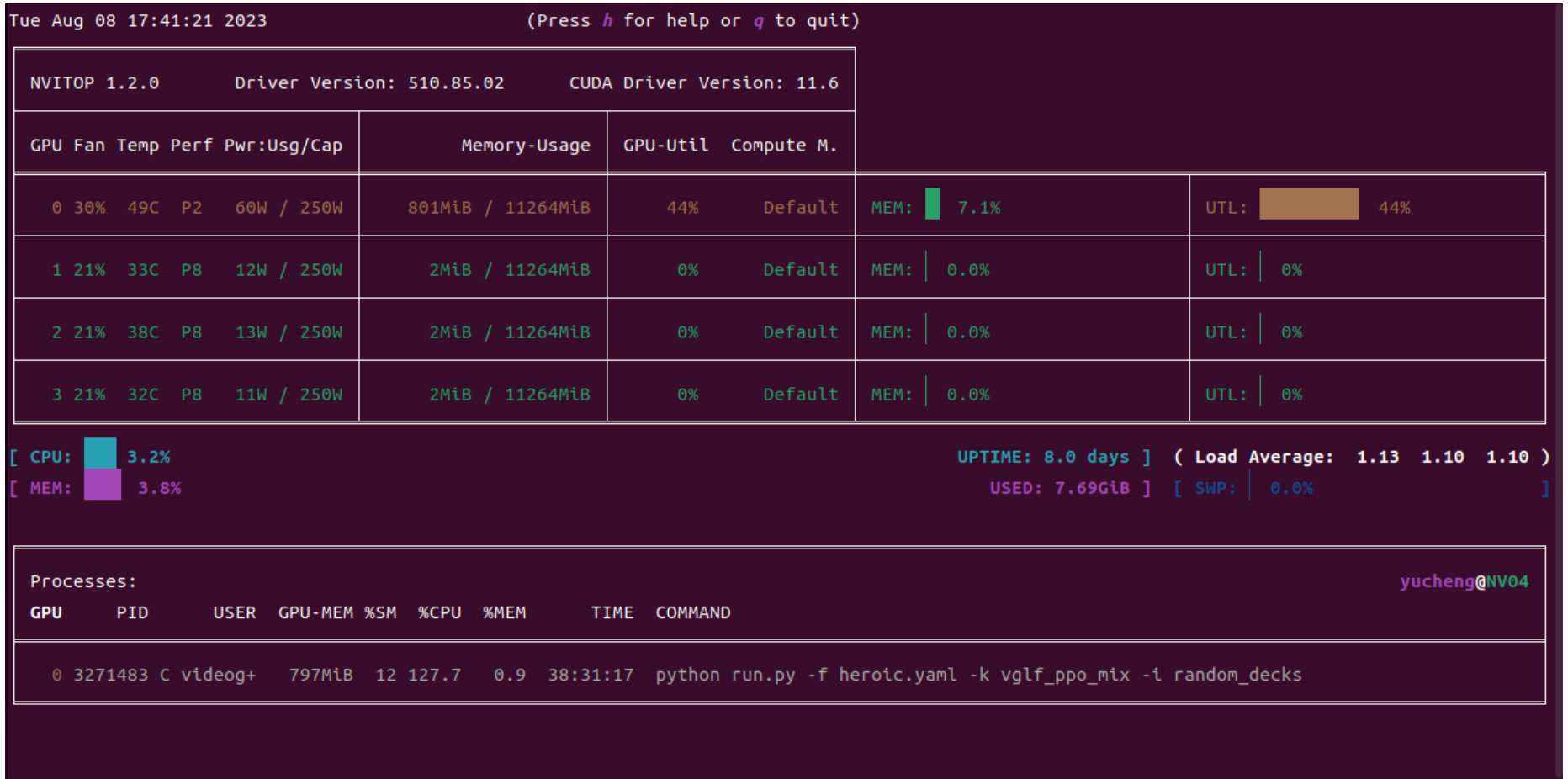
- apt install tmuxinator
- .tmuxinator.yml

```
name: test2
root: ~/

windows:
  - exercise:
      layout: even-horizontal
      panes:
        - p1: ping 8.8.8.8
        - p2: watch -n 2 -d nvidia-smi
```

- tmuxinator start .

Nvitop

- pip install nvitop



Q7 Associative array & sort

Write a bash script to calculate the frequency of each word in a text file `words.txt`.

For simplicity sake, you may assume:

- `words.txt` contains only lowercase characters and space `' '` characters.
- Each word must consist of lowercase characters only.
- Words are separated by one or more whitespace characters.

Example:

Assume that `words.txt` has the following content:

```
the day is sunny the the
the sunny is is
```

Your script should output the following, sorted by descending frequency:

```
the 4
is 3
sunny 2
day 1
```

```bash
#!/bin/bash

declare -A m

while read LINE
do
        line=$LINE
        for k in $line
        do
            m["$k"]=$((${m["$k"]}+1))
        done
done  < words.txt

                    @ also ok.
for key in ${!m[*]};
do
        echo $key ${m[$key]};
done | sort -r -k2 -n
```

Q8 awk

Given a text file `file.txt` that contains a list of phone numbers (one per line), write a one-liner bash script to print all valid phone numbers.

You may assume that a valid phone number must appear in one of the following two formats: (xxx) xxx-xxxx or xxx-xxx-xxxx. (x means a digit)

You may also assume each line in the text file must not contain leading or trailing white spaces.

Example:

Assume that `file.txt` has the following content:

```
987-123-4567
123 456 7890
(123) 456-7890
```

Your script should output the following valid phone numbers:

```
987-123-4567
(123) 456-7890
```

```
cat file.txt | awk
'$0 ~ /^([0-9][0-9][0-9])-[0-9][0-9][0-9]-[0-9][0-9][0-9][0-9]$|
^(\([0-9][0-9][0-9]\) )[0-9][0-9][0-9]-[0-9][0-9][0-9][0-9]$/ {print $0}'
```

Q9 awk, for, seq

Given a text file `file.txt`, transpose its content.

You may assume that each row has the same number of columns, and each field is separated by the `' '` character.

Example:

If `file.txt` has the following content:

```
name age
alice 21
ryan 30
```

Output the following:

```
name alice ryan
age 21 30
```

```
#!/bin/bash

                                                    c=2
c=$(cat file.txt | awk 'END {print NF}');

for i in $(seq 1 $c);
do
        k=$(cut -d' ' -f $i file.txt);
        echo $k;
done
```

Q10 awk

Given a text file `file.txt`, print just the 10th line of the file.

Example:

Assume that `file.txt` has the following content:

```
Line 1
Line 2
Line 3
Line 4
Line 5
Line 6
Line 7
Line 8
Line 9
Line 10
```

Your script should output the tenth line, which is:

```
Line 10
```

```bash
#!/bin/bash
i=0
while read LINE
do
        i=$(($i+1));
        if [ $i -eq 10 ]
        then
                echo $LINE
        fi
done  < file.txt
```

```bash
#!/bin/bash

cat file.txt | awk '{if (NR==10) print $0}'
```

Q11 change text color

- Change text color in terminal:

```
1 #!/bin/bash
2
3 while :
4 do
5   clear
6   for i in {1..20}
7   do
8     echo -n "*"
9     sleep 0.1
10  done
11  clear
12 done
13
14
```

```
1 #!/bin/bash
2
3 while :
4 do
5   clear
6   for i in {1..20}
7   do
8     echo -e -n "\033[0;32m*"
9     sleep 0.1
10  done
11  clear
12 done
13
```

enable interpretation of backslash escapes

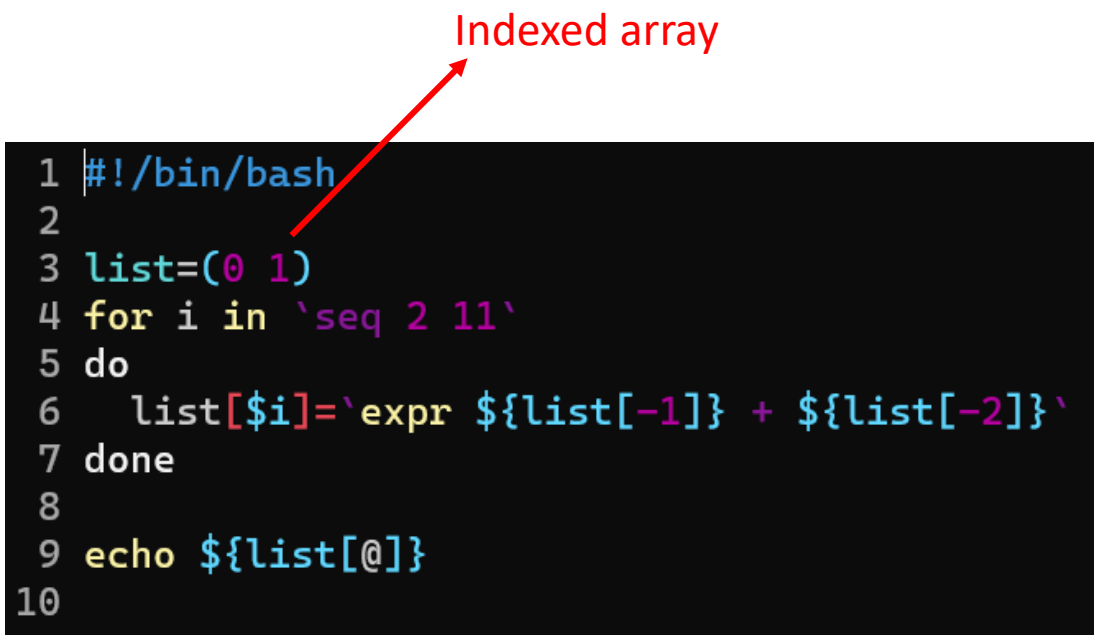
do not output the trailing newline

- Color list: https://gist.github.com/vratiu/9780109

Q12 array

- Access array elements and calculate:

```
1 #!/bin/bash
2
3 list=(0 1)
4 for i in `seq 2 11`
5 do
6   list[$i] = list[-1] + list[-2]
7 done
8
9 echo ${list[@]}
10
```

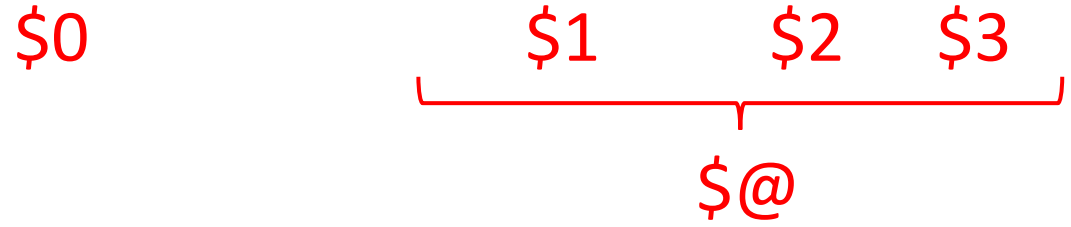


Q13 case

```bash
#!/bin/bash

ALL_PROCESS=$(ls /proc/ | egrep '[0-9]+')

running=0
stopped=0
sleeping=0
zombie=0

for pid in ${ALL_PROCESS[*]}
do
  test -f /proc/$pid/status && state=$(egrep "State" /proc/$pid/status | awk '{print $2}')
  case "$state" in
    R)
      running=$((running+1))
    ;;
    T)
      stopped=$((stopped+1))
    ;;
    S)
      sleeping=$((sleeping+1))
    ;;
    Z)
      zombie=$((zombie+1))
    ;;
  esac
done

echo -e "running: $running\nstopped: $stopped\nsleeping: $sleeping\nzombie: $zombie\n"
```

Q14 parse input arguments

- ./port_status_ans.sh 127.0.0.1 80 8080




```bash
#!/bin/bash

HOST=$1
PORT="22 25 80 8080"
for PORT in $PORT;
do
  if echo &>/dev/null > /dev/tcp/$HOST/$PORT; then
    echo "$PORT open"
  else
    echo "$PORT close"
  fi
done

```

```bash
#!/bin/bash

HOST=$1
PORT=$@
for PORT in ${PORT[*]};
do
  if [ "$PORT" = "$HOST" ]; then
    continue
  fi

  if echo &>/dev/null > /dev/tcp/$HOST/$PORT; then
    echo "$PORT open"
  else
    echo "$PORT close"
  fi
done

```

Q15 call function

String comparison needs “ ” both sides.

Function call like executing another script.

```bash
#!/bin/bash

function check_ip(){
  local IP=$1
  VALID_CHECK=$(echo $IP | awk -F. '$1<=255 && $2<=255 && $3<=255 && $4<=255 {print "yes"}')
  if echo $IP|grep -E "^[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}$" >/dev/null; then
    if [ $VALID_CHECK == "yes" ]; then
      return 0
    else
      echo "$IP not available!"
      return 1
    fi
  else
    echo "Format error! Please input again."
    return 1
  fi
}

while true;
do
    read -p "Please enter IP: " IP
    check_ip($IP)
    [ $? -eq 0 ] && break || continue
done

```

```bash
#!/bin/bash

function check_ip(){
  local IP=$1
  VALID_CHECK=$(echo $IP | awk -F. '$1<=255 && $2<=255 && $3<=255 && $4<=255 {print "yes"}')
  if echo $IP|grep -E "^[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}$" >/dev/null; then
    if [ "$VALID_CHECK" = "yes" ]; then
      return 0
    else
      echo "$IP not available!"
      return 1
    fi
  else
    echo "Format error! Please input again."
    return 1
  fi
}

while true;
do
    read -p "Please enter IP: " IP
    check_ip $IP
    [ $? -eq 0 ] && break || continue
done

```

Reference

- Awk:
  - Linux awk 命令 | 菜鸟教程 (runoob.com)
- Awk regular expression:
  - How to use regular expressions in awk | Opensource.com
- Nvidia PyTorch Released container images:
  - https://docs.nvidia.com/deeplearning/frameworks/pytorch-release-notes/rel-23-06.html
- Bash color list:
  - https://gist.github.com/vratiu/9780109
- Bash array:
  - https://www.hostinger.com/tutorials/bash-array
- Basic calculator (bc):
  - https://www.geeksforgeeks.org/bc-command-linux-examples/

Reference

- Shell script:
  - https://linux.vbird.org/linux_basic/centos7/0340bashshell-scripts.php
  - https://medium.com/@yihengwu/%E7%A8%8B%E5%BC%8F%E7%AD%86%E8%A8%98-shell-script-%E7%B0%A1%E6%98%93%E7%AD%86%E8%A8%98-841cfc3ae3ab
- Bash read:
  - https://www.delftstack.com/zh-tw/howto/linux/read-from-a-file-or-user-input-in-bash/
- Tmuxinator
  - https://github.com/tmuxinator/tmuxinator
- Tmux
  - https://blog.csdn.net/u014261408/article/details/89931729