Basic Linux and Slurm

“Linux and Slurm at the Command Line”

Weijun Gao @ UTSC Psychology

https://psycomp.utsc.utoronto.ca

About The Training…

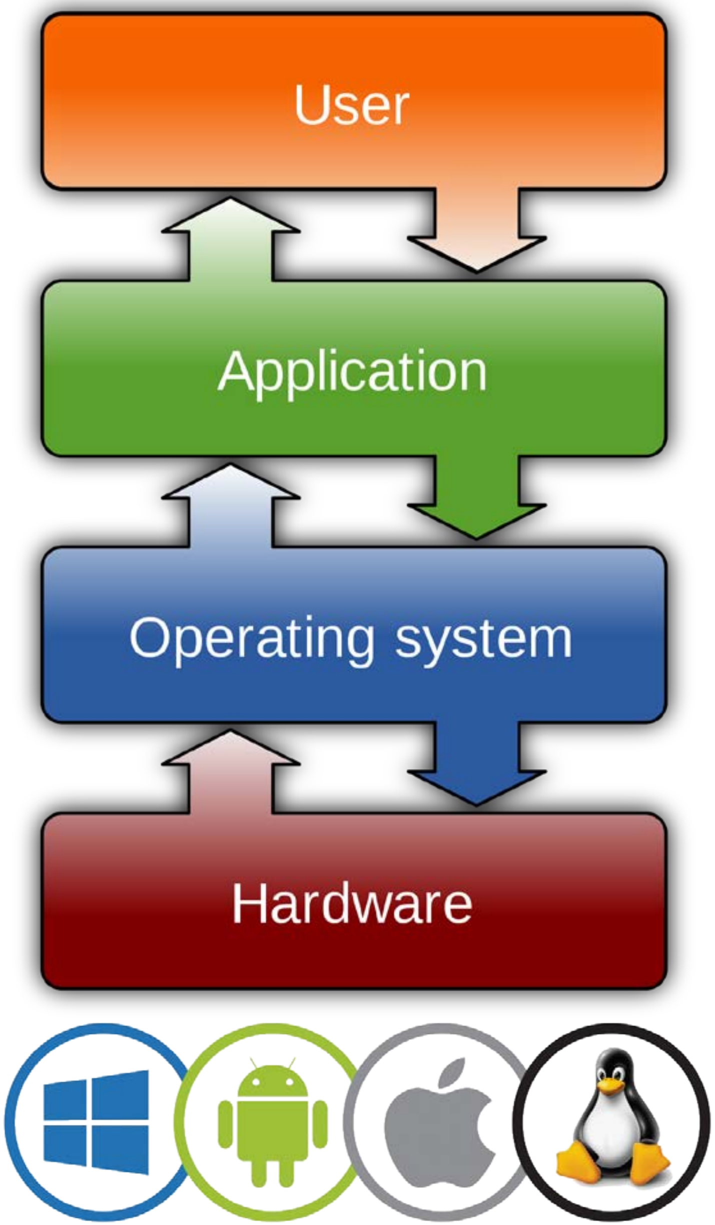
- What is Linux?
- What is a Computer Cluster
- What is Slurm?
- CLI vs GUI?
- Methods to Access Psy Cluster
- Obtaining the Examples
- Basic Linux
- BASH Script Examples
- Basic Slurm Concepts
- Basic Slurm Commands
- Slurm Examples
- More about Using Psy Cluster
- More Computing Support Topics …

What is Linux?

It’s an Operating System

It’s The Most Common Operating System Used By Researchers When Working on a Server or Computer Cluster

Free & Open Source



What is Linux?

- Linux is a Unix clone written from scratch by Linus Torvalds with assistance from a loosely-knit team of hackers across the world (www).
- Unix is a multitasking, multi-user computer operating system originally developed in 1969 by a group of AT&T employees at Bell Labs.
- Linux and Unix strive to be POSIX compliant.
- All top 500 super clusters are now running Linux. https://www.top500.org/statistics/details/osfam/1

What is Linux?

UNIX

Linux + GNU Utilities = GNU/Linux (Free Unix)

- Linux Kernel is an OS core written by Linus Torvalds and others. Linus' Minix became Linux.

https://www.kernel.org/

- GNU Utilities are a set of small programs written by Richard Stallman and others

http://www.gnu.org/

What is Linux?

Linux Has Many Distributions
https://distrowatch.com/

Psy Cluster is running Ubuntu server 18.04.

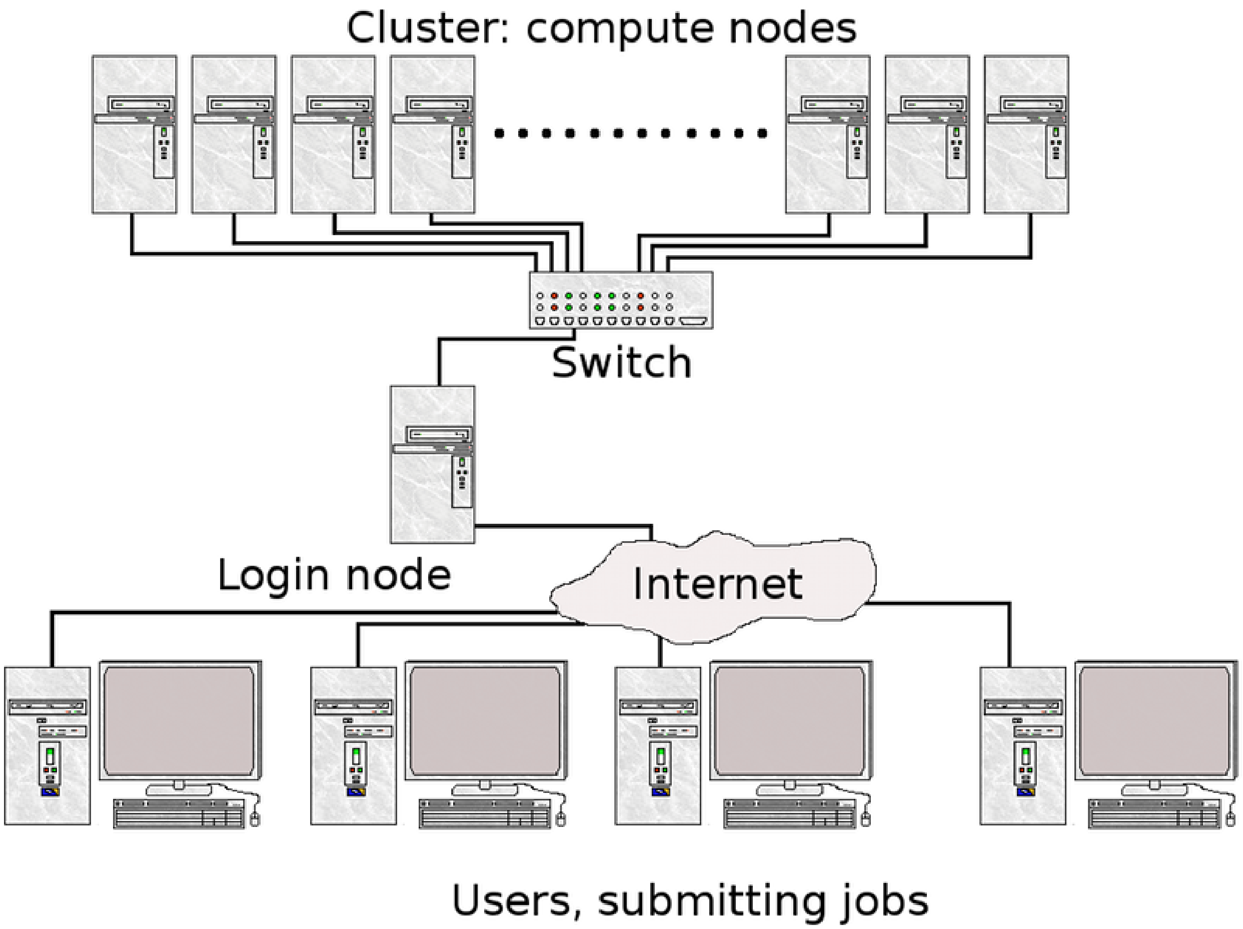
What is Linux?

- Debian (one of the few called GNU/Linux)
- Ubuntu (based on Debian)
- Slackware (one of the oldest, simple and stable distro)
- Redhat
  - RHEL (commercial support)
  - Fedora (free)
- CentOS (free RHEL)
- SuSE (OpenSuSE)
- Gentoo (Source code based)
- Knoppix (first LiveCD distro)
- …

Psy Cluster is running Ubuntu server 18.04.

What is a Computer Cluster?

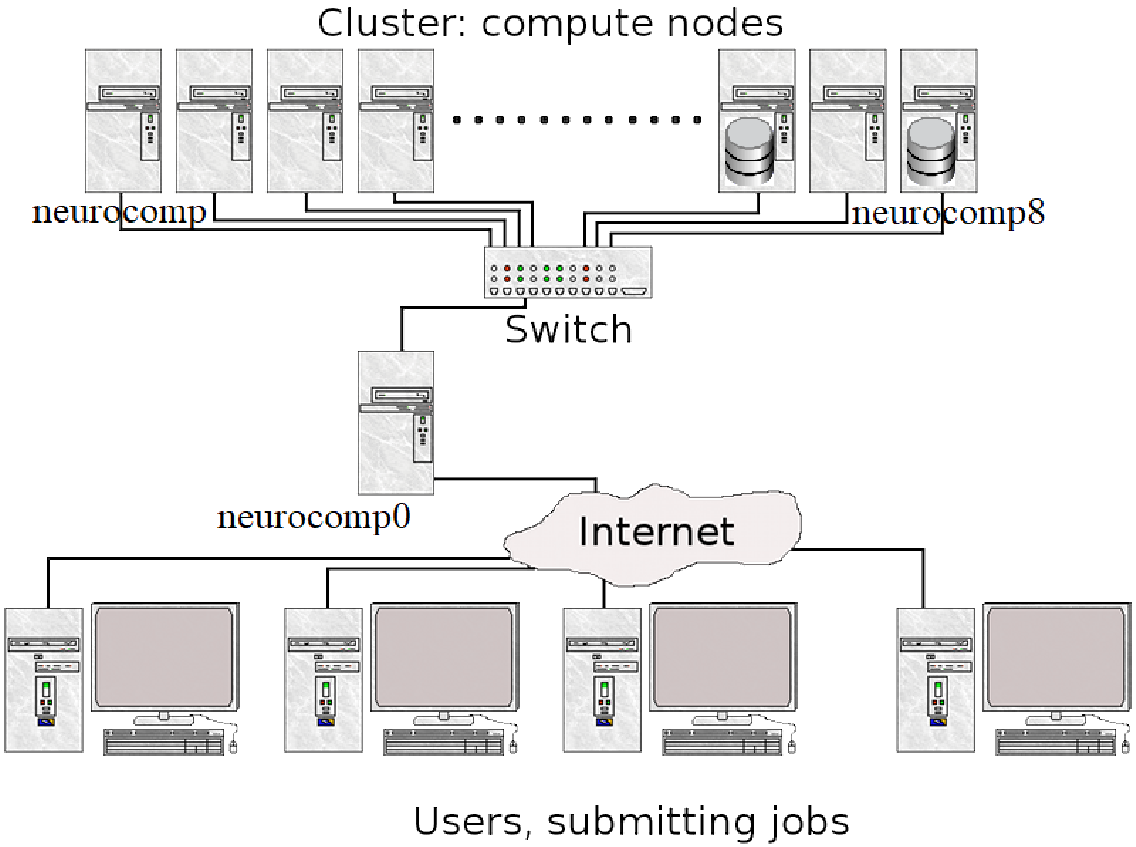
- A computer cluster is a single logical unit consisting of multiple computers that are linked through a LAN.
- The networked computers essentially act as a single, much more powerful machine.
- A cluster usually has head node(s) and compute node(s).
- Computer clusters have each compute node configured and controlled the same way by software, not hardware.



What is a Computer Cluster?

Psy Cluster

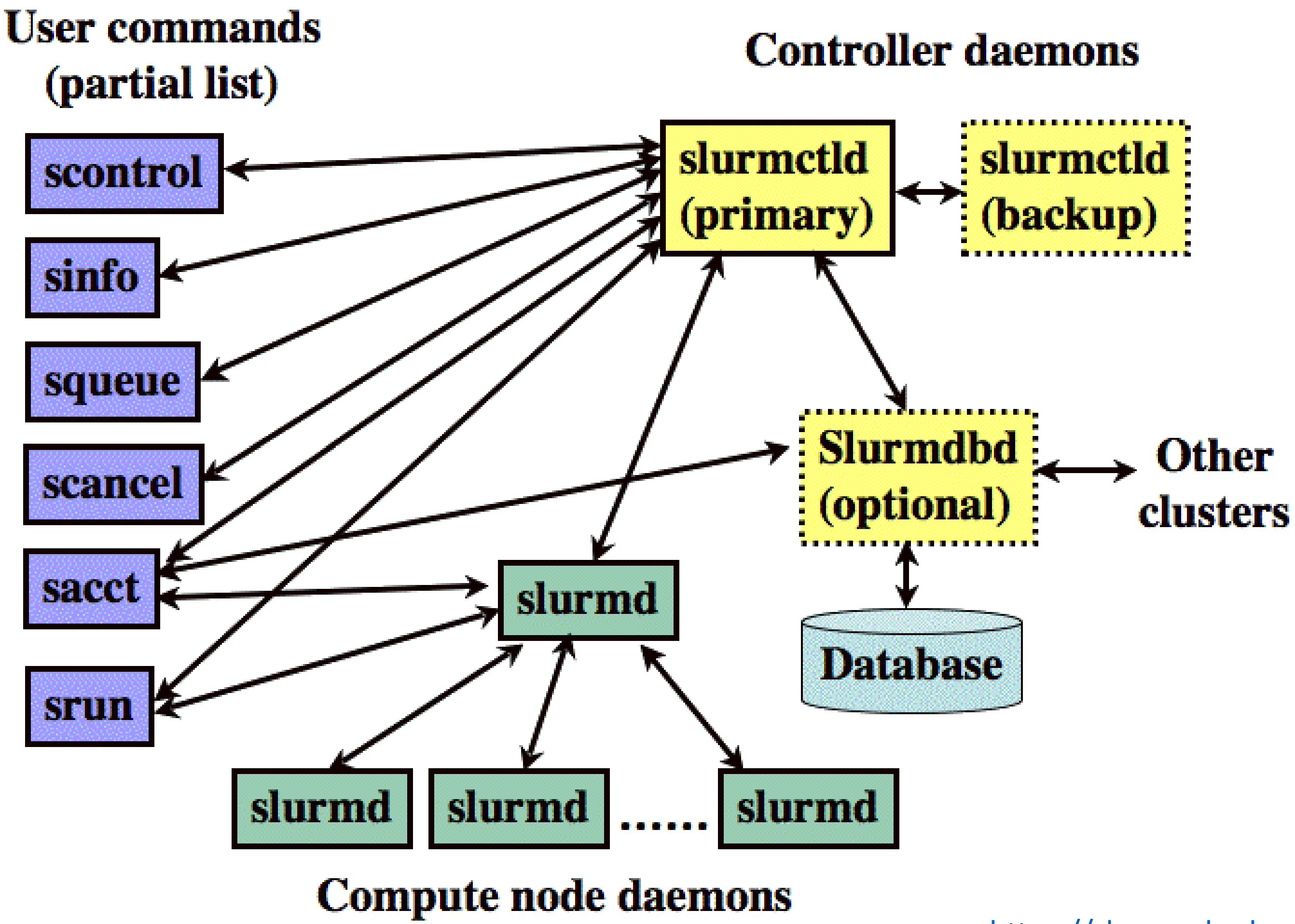
- Head node(s) – neurocomp0
- Compute node(s) – neurocomp, neurocomp[2-8],…
- Storage - /psyhome6, /psyhome8
- Operating System – Ubuntu 18.04
- Job scheduler - Slurm



What is Slurm?

- Slurm (Simple Linux Utility for Resource Management or SLURM) is a job scheduler for Linux or Unix-like systems
- Surlm is free, open-source, fault-tolerant and highly scalable
- Slurm allocates exclusive and/or non-exclusive access to resources to users for some duration of time.
- Slurm provides a framework for starting, executing, and monitoring (parallel) jobs.
- Slurm arbitrates contention for resources by managing a queue of pending jobs.
- Over 60% top 500 super clusters use Slurm.

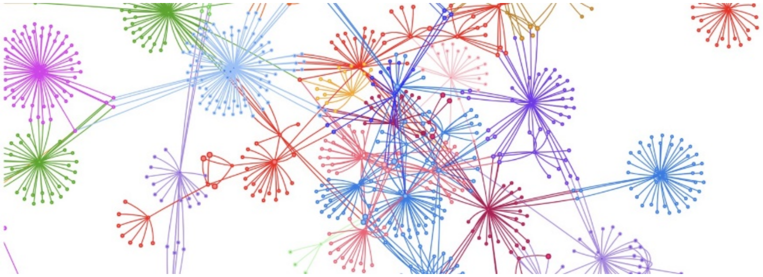
What is Slurm?

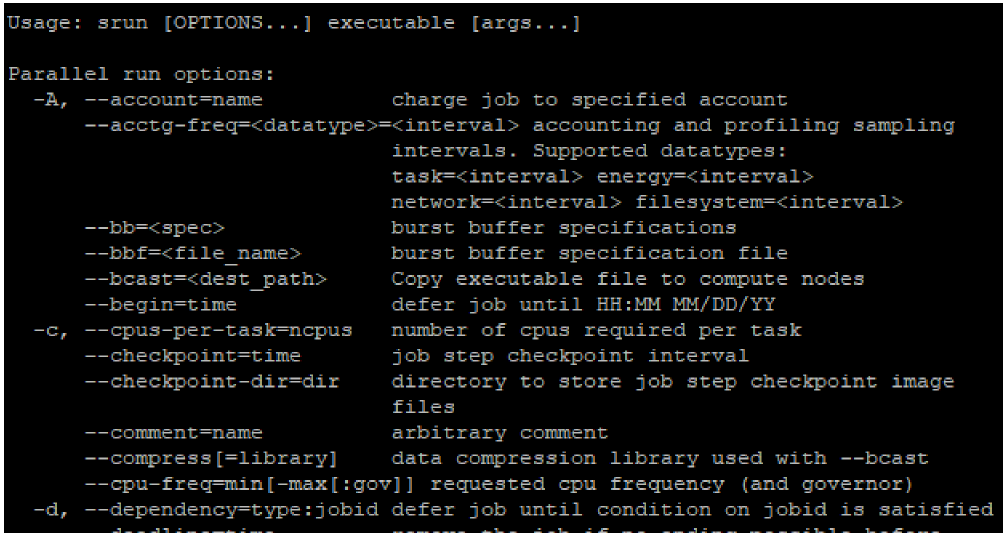


https://slurm.schedmd.com/arch.gif

CLI vs GUI

- CLI – Command Line Interface
- GUI – Graphic User Interface
- CLI – learning curve
- GUI – easy to use
- CLI – faster
- CLI – more powerful
- CLI – more efficient
- CLI – repeatable (exact)
- CLI – programmable
- CLI – automation
- GUI – visualization





```
Usage: srun [OPTIONS...] executable [args...]

Parallel run options:
  -A, --account=name          charge job to specified account
      --acctg-freq=<datatype>=<interval> accounting and profiling sampling
                              intervals. Supported datatypes:
                              task=<interval> energy=<interval>
                              network=<interval> filesystem=<interval>
      --bb=<spec>             burst buffer specifications
      --bbf=<file_name>       burst buffer specification file
      --bcast=<dest_path>     Copy executable file to compute nodes
      --begin=time            defer job until HH:MM MM/DD/YY
  -c, --cpus-per-task=ncpus   number of cpus required per task
      --checkpoint=time       job step checkpoint interval
      --checkpoint-dir=dir    directory to store job step checkpoint image
                              files
      --comment=name          arbitrary comment
      --compress[=library]    data compression library used with --bcast
      --cpu-freq=min[-max[:gov]] requested cpu frequency (and governor)
  -d, --dependency=type:jobid defer job until condition on jobid is satisfied
```

Methods to Access Psy Cluster

- X2Go — NOT for Rstudio users
- SSH — (Mac OS X is Unix)
- PuTTY — For X support
- Git BASH

VcXsrv Windows X Server

~~Xming~~

Psy Cluster Headnode: neurocomp0.utsc.utoronto.ca

Methods to Access Psy Cluster

- FileZilla
- scp / rsync
- PSCP / cwRsync / ~~WinSCP~~

Psy Cluster Head Node: neurocomp0.utsc.utoronto.ca

Methods to Access Psy Cluster

- Connection problems?

```
ping neurocomp0.utsc.utoronto.ca
telnet neurocomp0.utsc.utoronto.ca 22
```

Network speed test: https://speedtest.utoronto.ca/
Cluster Status: https://psycomp.utsc.utoronto.ca/

Psy Cluster Head Node: neurocomp0.utsc.utoronto.ca

Obtaining the Example Scripts

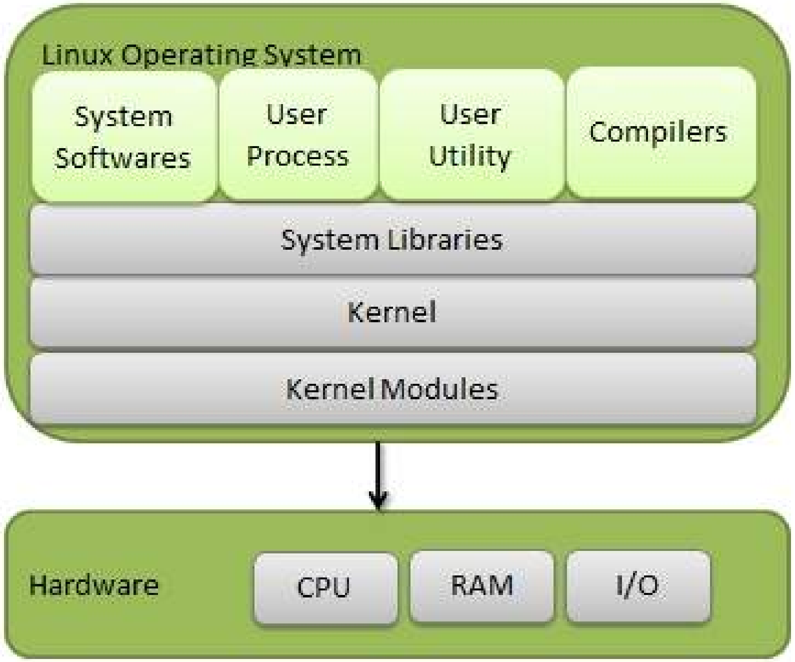
https://psycomp.utsc.utoronto.ca/support/index.php/resources/training-materials/

Visit Computing Support site:

> COMPUTING & RESOURCES > TRAINING MATERIALS

Psy Cluster Computing Support: https://psycomp.utsc.utoronto.ca/

Basic Linux



Applications

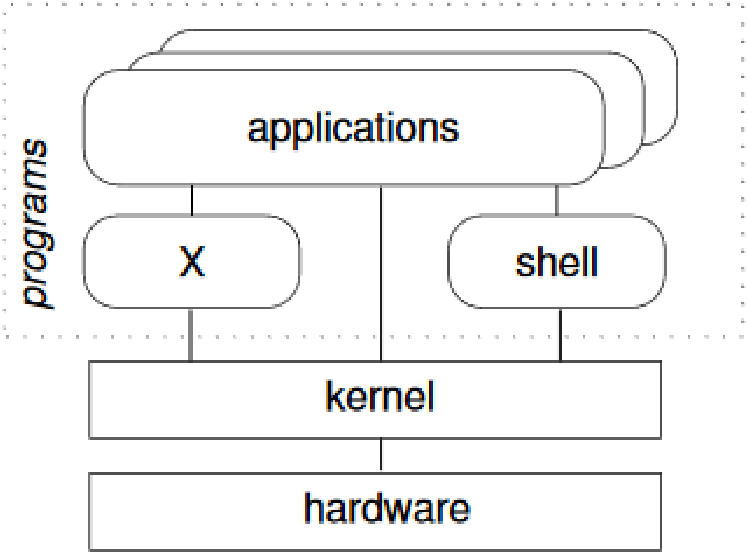
Middleware

Operating System

Drivers

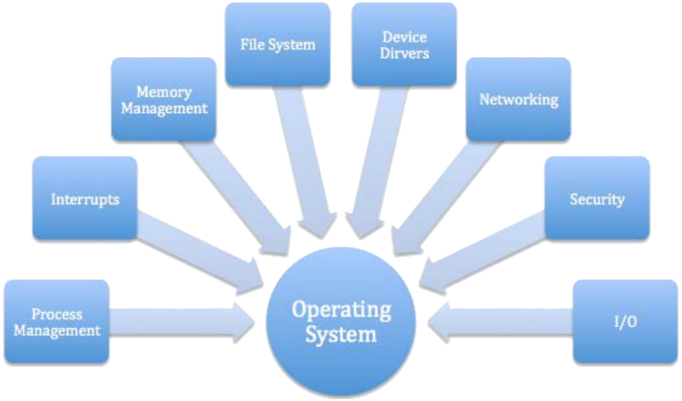
Firmware **(important - firmware update)**

Hardware

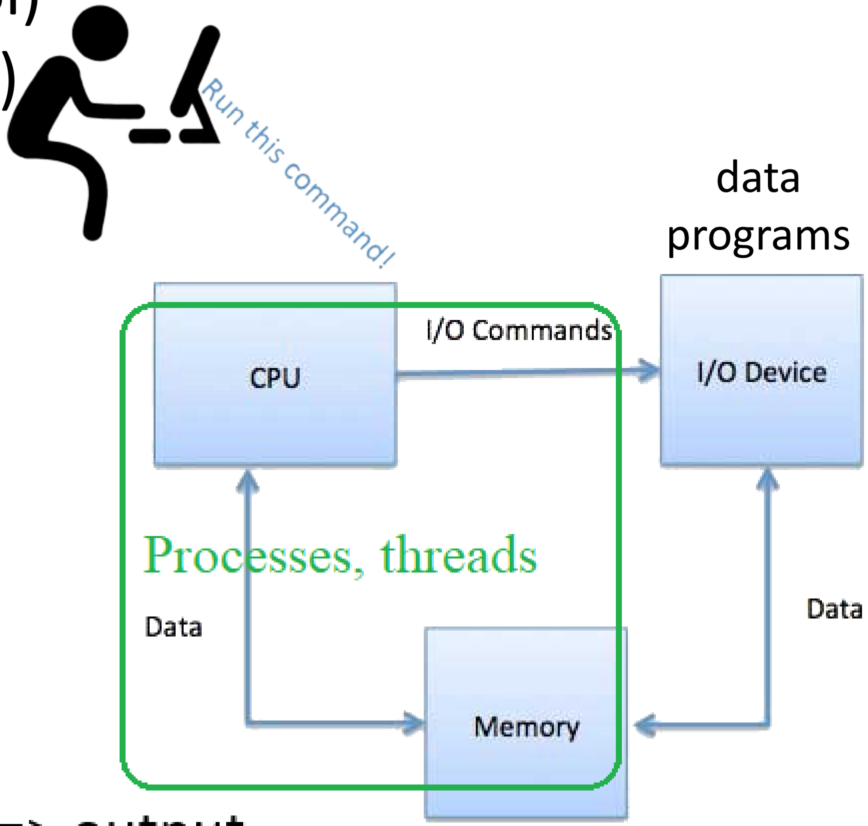


input => **program** => output

Basic Linux



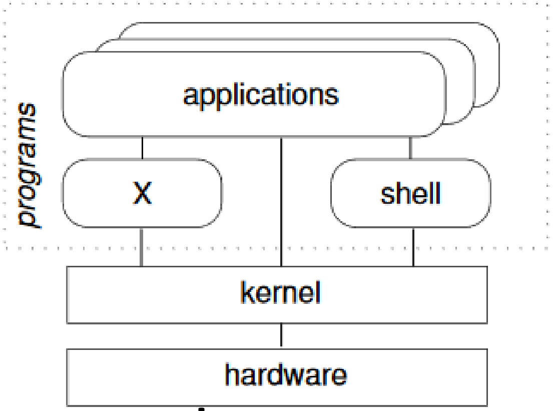
- Users (accounts, access control)
- Resources (CPU, storage, RAM)
- Storage (data, programs)
- Applications / Programs
- Processes, Threads
- Services
- Drivers
- Networking
- I/O
- …



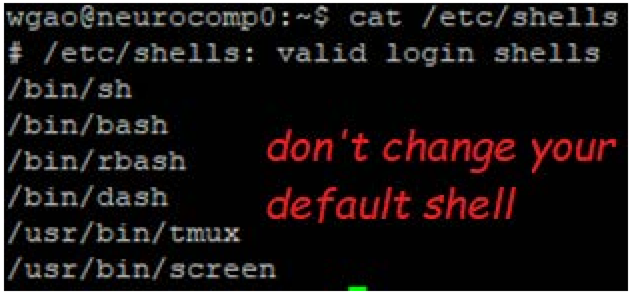
input => program => output

Basic Linux

"Small programs that do one thing well"



- A shell is a computer program that interprets the commands you type and sends them to the operating system;
- A shell provides a powerful programming environment, capable of automating nearly anything on a Linux system;
- A shell = commands + variables + syntax: bash; csh; tcsh; …
- Change shell: *chsh -s /bin/bash*



```
wgao@neurocomp0:~$ cat /etc/shells
# /etc/shells: valid login shells
/bin/sh
/bin/bash
/bin/rbash
/bin/dash
/usr/bin/tmux
/usr/bin/screen
```

don't change your default shell

Basic Linux

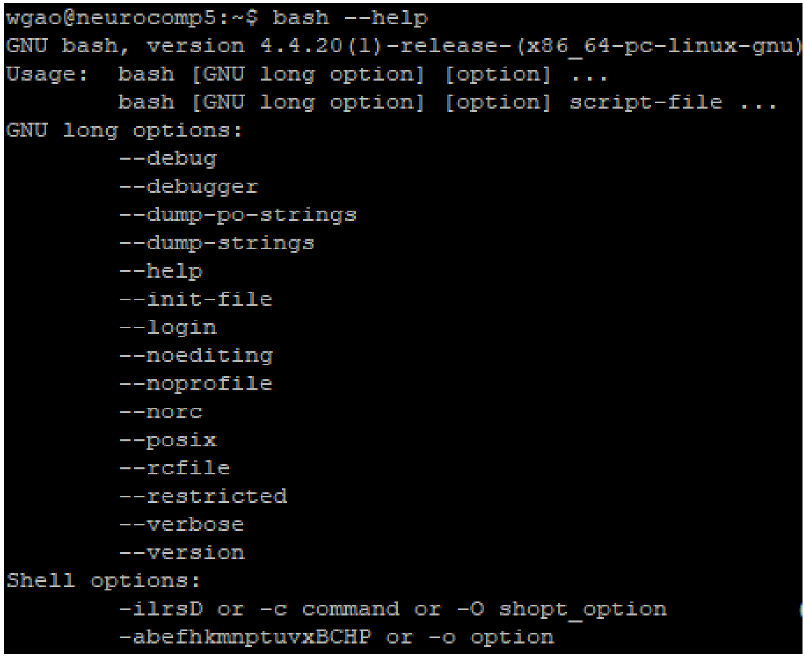
“Small programs that do one thing well”

- Command line arguments / parameters
- Help: man CMD; info CMD; CMD --help

  man ls

  info df

  srun --help



```
wgao@neurocomp5:~$ bash --help
GNU bash, version 4.4.20(1)-release-(x86_64-pc-linux-gnu)
Usage:  bash [GNU long option] [option] ...
        bash [GNU long option] [option] script-file ...
GNU long options:
        --debug
        --debugger
        --dump-po-strings
        --dump-strings
        --help
        --init-file
        --login
        --noediting
        --noprofile
        --norc
        --posix
        --rcfile
        --restricted
        --verbose
        --version
Shell options:
        -ilrsD or -c command or -O shopt_option
        -abefhkmnptuvxBCHP or -o option
```

- File System

  ls; cd; pwd; tree; find; grep; which

  cat; head; tail; less; more; diff

  mv; cp; ln; rm; mkdir; rmdir; touch

  chmod

  df; du

Basic Linux

“Small programs that do one thing well”

- Pipe and redirection: |; > ; >>; <; &>; 2>; 2>&1; |&

  Display only files: *ls -l | grep -v '^d'*

  Read input from a file instead of stdin: *cat < test.sh*

  Redirect stdout and stderr to a file: *ls -l &> ls.txt; cat ls.txt*

- Environment variables and conf files

  env

  set

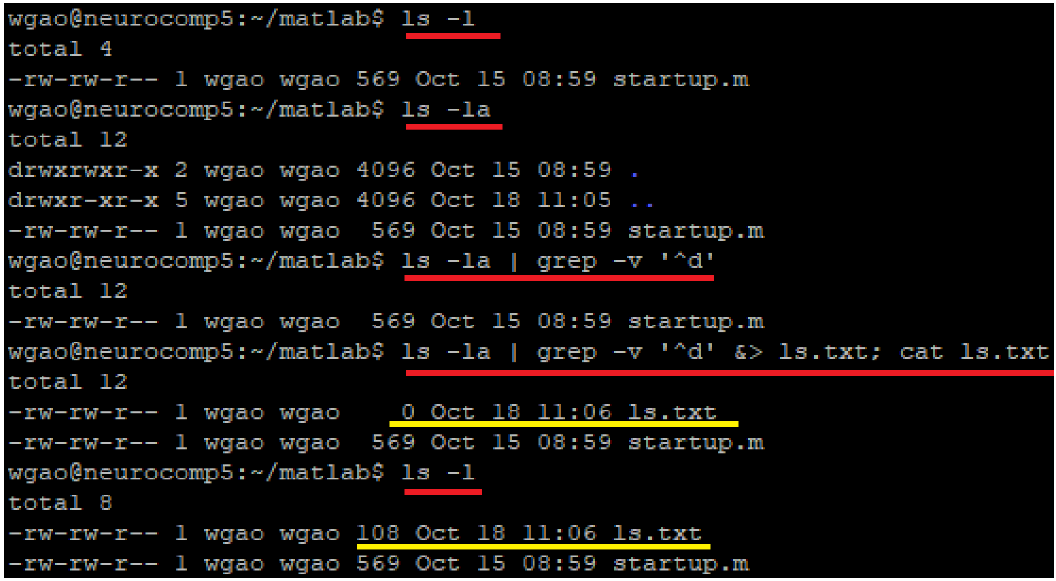
  export PATH=$PATH:~/bin

  echo $PATH

  echo $HOME

  which python

  cat ~/.profile



```
wgao@neurocomp5:~/matlab$ ls -l
total 4
-rw-rw-r-- 1 wgao wgao 569 Oct 15 08:59 startup.m
wgao@neurocomp5:~/matlab$ ls -la
total 12
drwxrwxr-x 2 wgao wgao 4096 Oct 15 08:59 .
drwxr-xr-x 5 wgao wgao 4096 Oct 18 11:05 ..
-rw-rw-r-- 1 wgao wgao  569 Oct 15 08:59 startup.m
wgao@neurocomp5:~/matlab$ ls -la | grep -v '^d'
total 12
-rw-rw-r-- 1 wgao wgao  569 Oct 15 08:59 startup.m
wgao@neurocomp5:~/matlab$ ls -la | grep -v '^d' &> ls.txt; cat ls.txt
total 12
-rw-rw-r-- 1 wgao wgao    0 Oct 18 11:06 ls.txt
-rw-rw-r-- 1 wgao wgao  569 Oct 15 08:59 startup.m
wgao@neurocomp5:~/matlab$ ls -l
total 8
-rw-rw-r-- 1 wgao wgao 108 Oct 18 11:06 ls.txt
-rw-rw-r-- 1 wgao wgao 569 Oct 15 08:59 startup.m
```

Basic Linux

free, open-source & inevitable bugs

- Hotkeys

  Cancel a running interactive process: <ctrl><c>, <ctrl><z>

  Navigate bash command history: up / down arrows (↑↓)

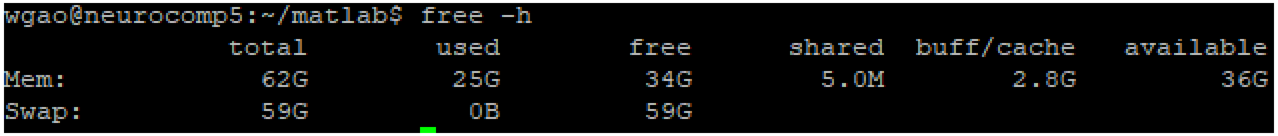
- Regular Expression

  Linux users often need to use regular expression to search in files and output

  grep

- System status

  df, free, top, htop, ps …



```
wgao@neurocomp5:~/matlab$ free -h
              total        used        free      shared  buff/cache   available
Mem:            62G         25G         34G        5.0M        2.8G         36G
Swap:           59G          0B         59G
```

Basic Linux

free, open-source & inevitable bugs

- Run programs in background: &

  cp -pqr ~/folder1 ~/folder2 &

- Keep an SSH session alive

  screen

  tmux

- Text editors

  gedit, leafpad, emacs

  vim / vi

- Data transfer

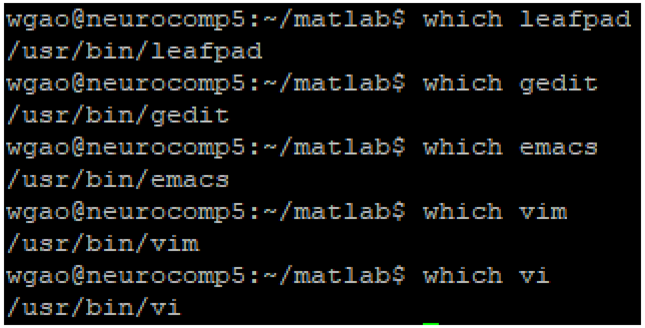
  scp / pscp

  rsync / cwRsync

  FileZilla

  WinSCP

  cp / mv

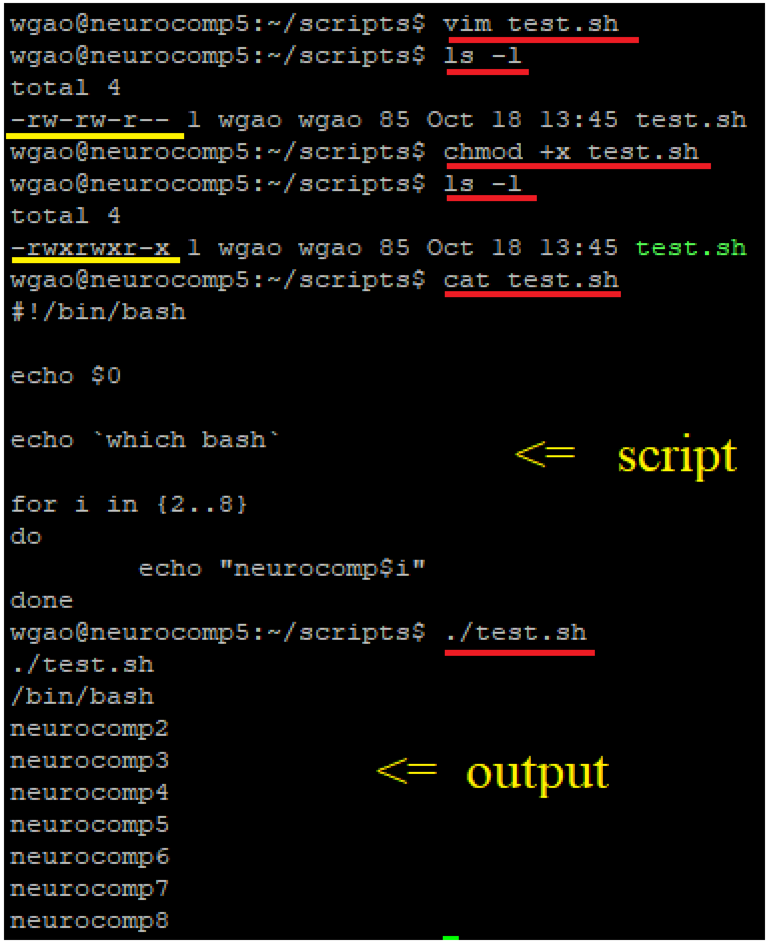


```
wgao@neurocomp5:~/matlab$ which leafpad
/usr/bin/leafpad
wgao@neurocomp5:~/matlab$ which gedit
/usr/bin/gedit
wgao@neurocomp5:~/matlab$ which emacs
/usr/bin/emacs
wgao@neurocomp5:~/matlab$ which vim
/usr/bin/vim
wgao@neurocomp5:~/matlab$ which vi
/usr/bin/vi
```

Basic Linux

free, more secure, stable and flexible

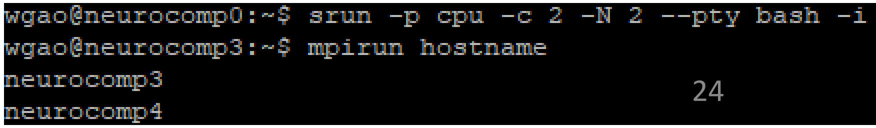
- Process management
  - *ps, top, htop*
  - *kill, killall*
  - *fg, bg*
- Shell scripting
  - $#
  - $0, $1, …
  - logic and loops
- Parallel computing
  - MPI (multiple nodes)
  - OpenMP (within a node)
  - Processes & Threads



```
wgao@neurocomp5:~/scripts$ vim test.sh
wgao@neurocomp5:~/scripts$ ls -l
total 4
-rw-rw-r-- 1 wgao wgao 85 Oct 18 13:45 test.sh
wgao@neurocomp5:~/scripts$ chmod +x test.sh
wgao@neurocomp5:~/scripts$ ls -l
total 4
-rwxrwxr-x 1 wgao wgao 85 Oct 18 13:45 test.sh
wgao@neurocomp5:~/scripts$ cat test.sh
#!/bin/bash

echo $0

echo `which bash`

for i in {2..8}
do
        echo "neurocomp$i"
done
wgao@neurocomp5:~/scripts$ ./test.sh
./test.sh
/bin/bash
neurocomp2
neurocomp3
neurocomp4
neurocomp5
neurocomp6
neurocomp7
neurocomp8
```

<= script

<= output



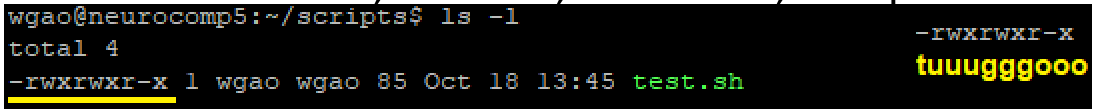
```
wgao@neurocomp0:~$ srun -p cpu -c 2 -N 2 --pty bash -i
wgao@neurocomp3:~$ mpirun hostname
neurocomp3
neurocomp4
```

Basic Linux

free, more secure, stable and flexible

- Access control

r:read, w:write, x:execute, -:no permissions

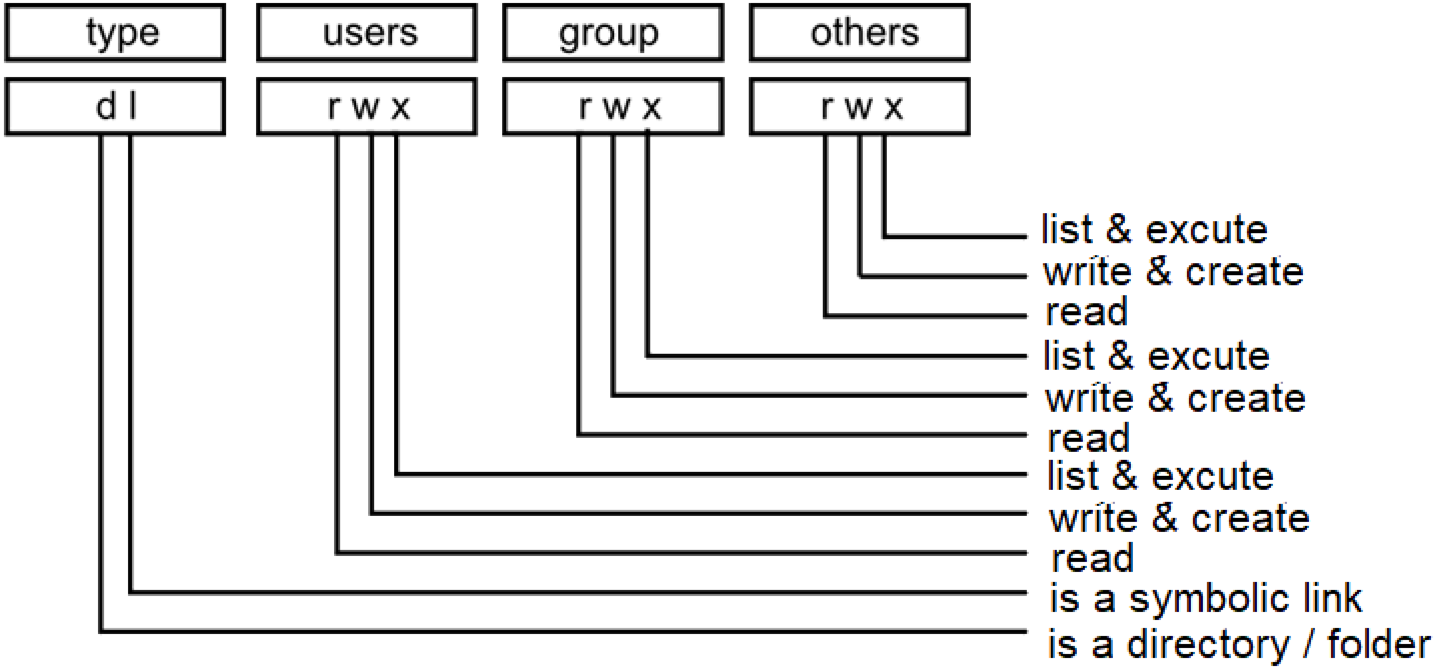


```
wgao@neurocomp5:~/scripts$ ls -l
total 4
-rwxrwxr-x 1 wgao wgao 85 Oct 18 13:45 test.sh
```

-rwxrwxr-x
tuuugggooo

chmod 0777 ./tmp

octal 7 is binary 111 (rwx)



type
users
group
others

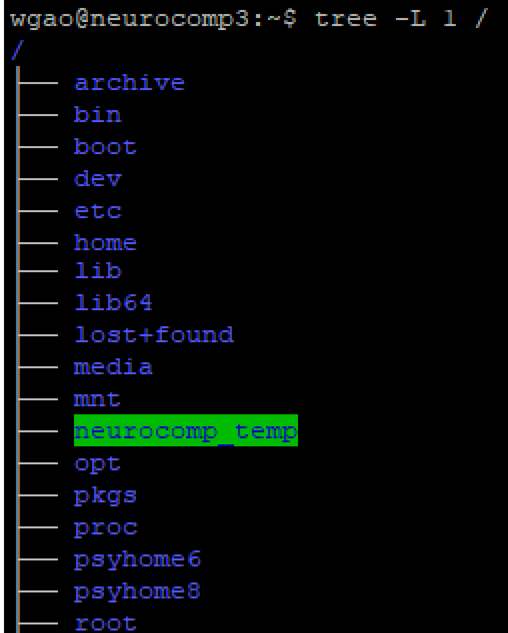
d l
r w x
r w x
r w x

list & excute
write & create
read
list & excute
write & create
read
list & excute
write & create
read
is a symbolic link
is a directory / folder

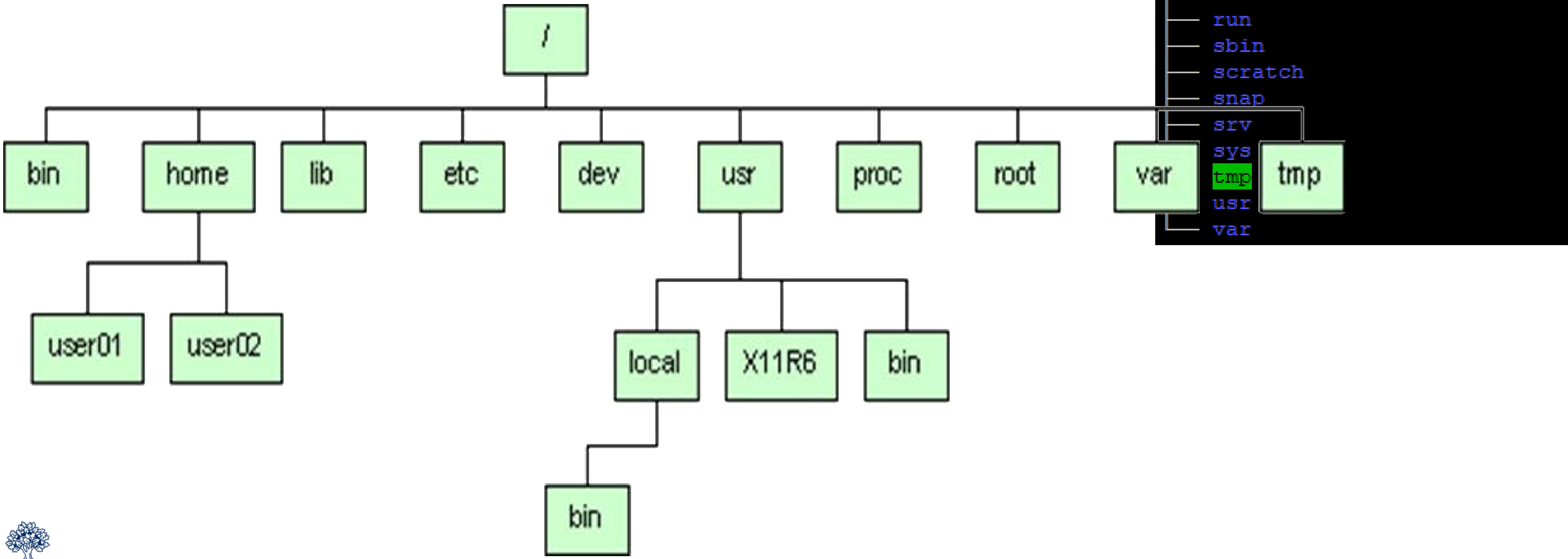
Basic Linux

free, more secure, stable and flexible

- Directory structure (tree)

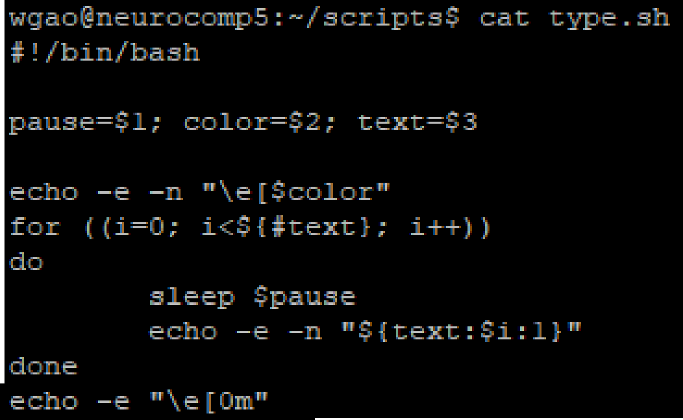


```
wgao@neurocomp3:~$ tree -L 1 /
/
├── archive
├── bin
├── boot
├── dev
├── etc
├── home
├── lib
├── lib64
├── lost+found
├── media
├── mnt
├── neurocomp_temp
├── opt
├── pkgs
├── proc
├── psyhome6
├── psyhome8
├── root
├── run
├── sbin
├── scratch
├── snap
├── srv
├── sys
├── tmp
├── usr
└── var
```



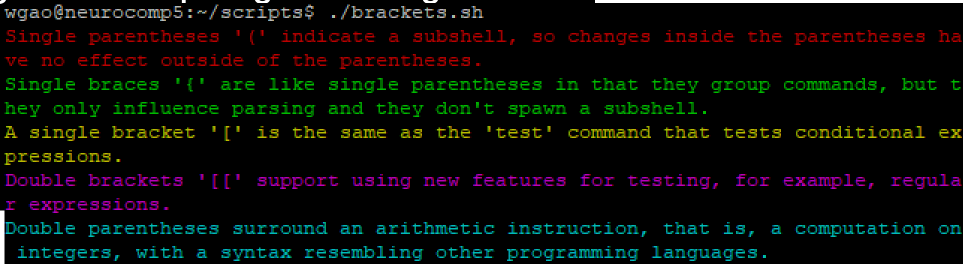
BASH Script Examples

square brackets [], parentheses () and braces {} explanations in colors



```
wgao@neurocomp5:~/scripts$ cat type.sh
#!/bin/bash

pause=$1; color=$2; text=$3

echo -e -n "\e[$color"
for ((i=0; i<${#text}; i++))
do
        sleep $pause
        echo -e -n "${text:$i:1}"
done
echo -e "\e[0m"
```

```
#!/bin/bash

# this script calls “./type.sh” to simulate keyboard typing

./type.sh 0.1 31m "Single parentheses '(' indicate a subshell, so changes inside the parentheses have no effect outside of the parentheses."

./type.sh 0.1 32m "Single braces '{' are like single parentheses in that they group commands, but they only influence parsing and they don't spawn a subshell."

./type.sh 0.1 33m "A single bracket '[' is the same as the 'test' command that tests conditional expressions."

./type.sh 0.1 35m "Double brackets '[[' support using new features for testing, for example, regular expressions."

./type.sh 0.1 36m "Double parentheses surround an arithmetic instruction, that is, a computation on integers, with a syntax resembling other programming languages."
```

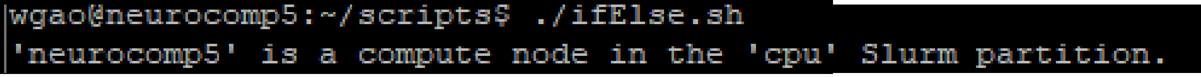


```
wgao@neurocomp5:~/scripts$ ./brackets.sh
Single parentheses '(' indicate a subshell, so changes inside the parentheses ha
ve no effect outside of the parentheses.
Single braces '{' are like single parentheses in that they group commands, but t
hey only influence parsing and they don't spawn a subshell.
A single bracket '[' is the same as the 'test' command that tests conditional ex
pressions.
Double brackets '[[' support using new features for testing, for example, regula
r expressions.
Double parentheses surround an arithmetic instruction, that is, a computation on
 integers, with a syntax resembling other programming languages.
```

BASH Script Examples

if-else statement

```
#!/bin/bash

hostname=`hostname`

echo -n "'$hostname' is "
if [[ "$hostname" == "neurocomp" ]]; then
        type="a backup head node and a compute node in the 'interactive' Slurm partition"
elif [[ "$hostname" == "neurocomp6" ]]; then
        type="a compute node in the 'interactive' Slrum partition"
elif [[ $hostname =~ ^neurocomp(0|00|-teach)$ ]]; then
        type="a head node"
elif [[ $hostname =~ ^neurocomp[2-8]$ ]]; then
        type="a compute node in the 'cpu' Slurm partition"
else
        type="an unknown host"
fi
echo "$type."
```

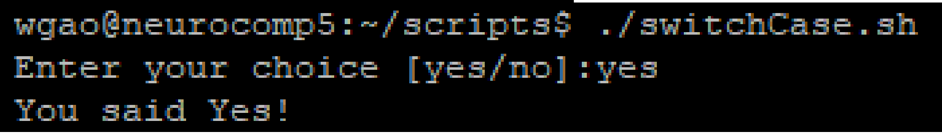


```
wgao@neurocomp5:~/scripts$ ./ifElse.sh
'neurocomp5' is a compute node in the 'cpu' Slurm partition.
```

BASH Script Examples

switch-case statement

```
#!/bin/bash

# read from standard input to variable “choice”
read -p "Enter your choice [yes/no]:" choice

# make “choice” string all lower case
choice=`echo -n "$choice" | tr '[:upper:]' '[:lower:]'`

case $choice in
        yes)
                echo "You said Yes!"
                ;;
        no)
                echo "You said No!"
                ;;
        *)
                echo "Please enter 'yes' or 'no'."
                ;;
esac
```

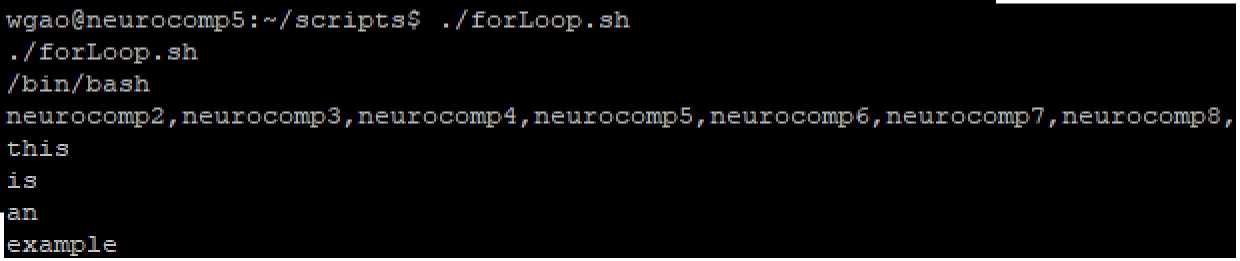


```
wgao@neurocomp5:~/scripts$ ./switchCase.sh
Enter your choice [yes/no]:yes
You said Yes!
```

BASH Script Examples

for-loop

```
#!/bin/bash

echo $0

echo `which bash`

for i in {2..8}
do
        echo -n "neurocomp$i,"
done

echo

for i in this is an example
do
        echo $i
done
```

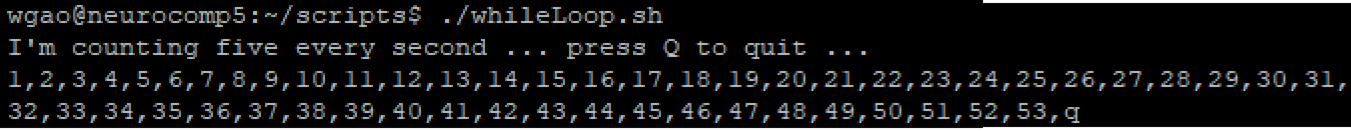


```
wgao@neurocomp5:~/scripts$ ./forLoop.sh
./forLoop.sh
/bin/bash
neurocomp2,neurocomp3,neurocomp4,neurocomp5,neurocomp6,neurocomp7,neurocomp8,
this
is
an
example
```

BASH Script Examples

while-loop

```
#!/bin/bash

echo "I'm counting five every second ... press Q to quit ..."

count=1

while true;
do
        echo -n "$count,"
        # read one character (-N 1), times out after 0.2 seconds (-t 0.2)
        read -t 0.2 -N 1 input
        count=$((count + 1))
        if [[ $input =~ ^(q|Q)$ ]]; then
                echo
                break;
        fi
done
```

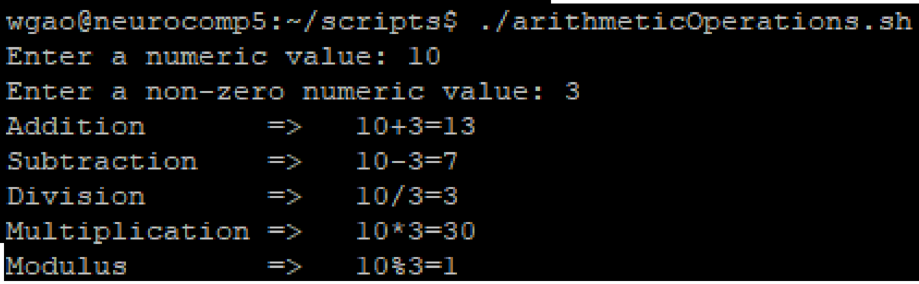


```
wgao@neurocomp5:~/scripts$ ./whileLoop.sh
I'm counting five every second ... press Q to quit ...
1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,
32,33,34,35,36,37,38,39,40,41,42,43,44,45,46,47,48,49,50,51,52,53,q
```

BASH Script Examples

arithmetic operations

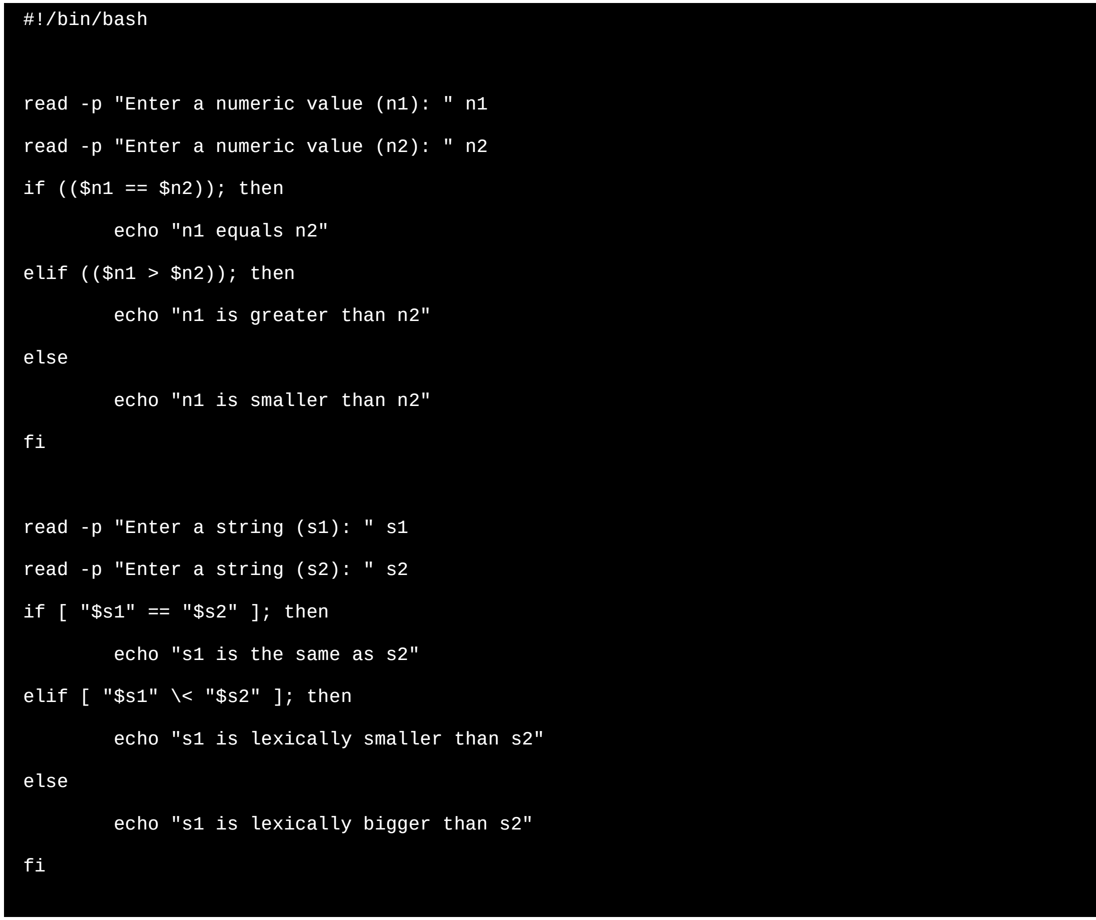
```
#!/bin/bash

read -p "Enter a numeric value: " n1
read -p "Enter a non-zero numeric value: " n2

echo "Addition       =>   $n1+$n2="$((n1+n2))
echo "Subtraction    =>   $n1-$n2="$((n1-n2))
echo "Division       =>   $n1/$n2="$((n1/n2))
echo "Multiplication =>   $n1*$n2="$((n1*n2))
echo "Modulus        =>   $n1%$n2="$((n1%n2))
```



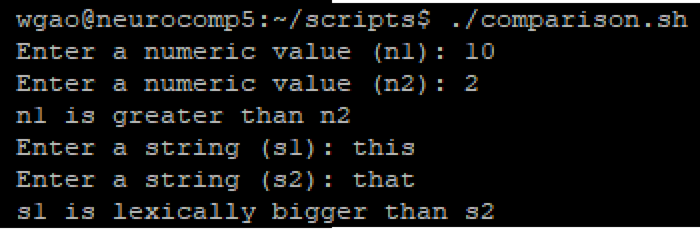
```
wgao@neurocomp5:~/scripts$ ./arithmeticOperations.sh
Enter a numeric value: 10
Enter a non-zero numeric value: 3
Addition       =>   10+3=13
Subtraction    =>   10-3=7
Division       =>   10/3=3
Multiplication =>   10*3=30
Modulus        =>   10%3=1
```

BASH Script Examples

comparisons



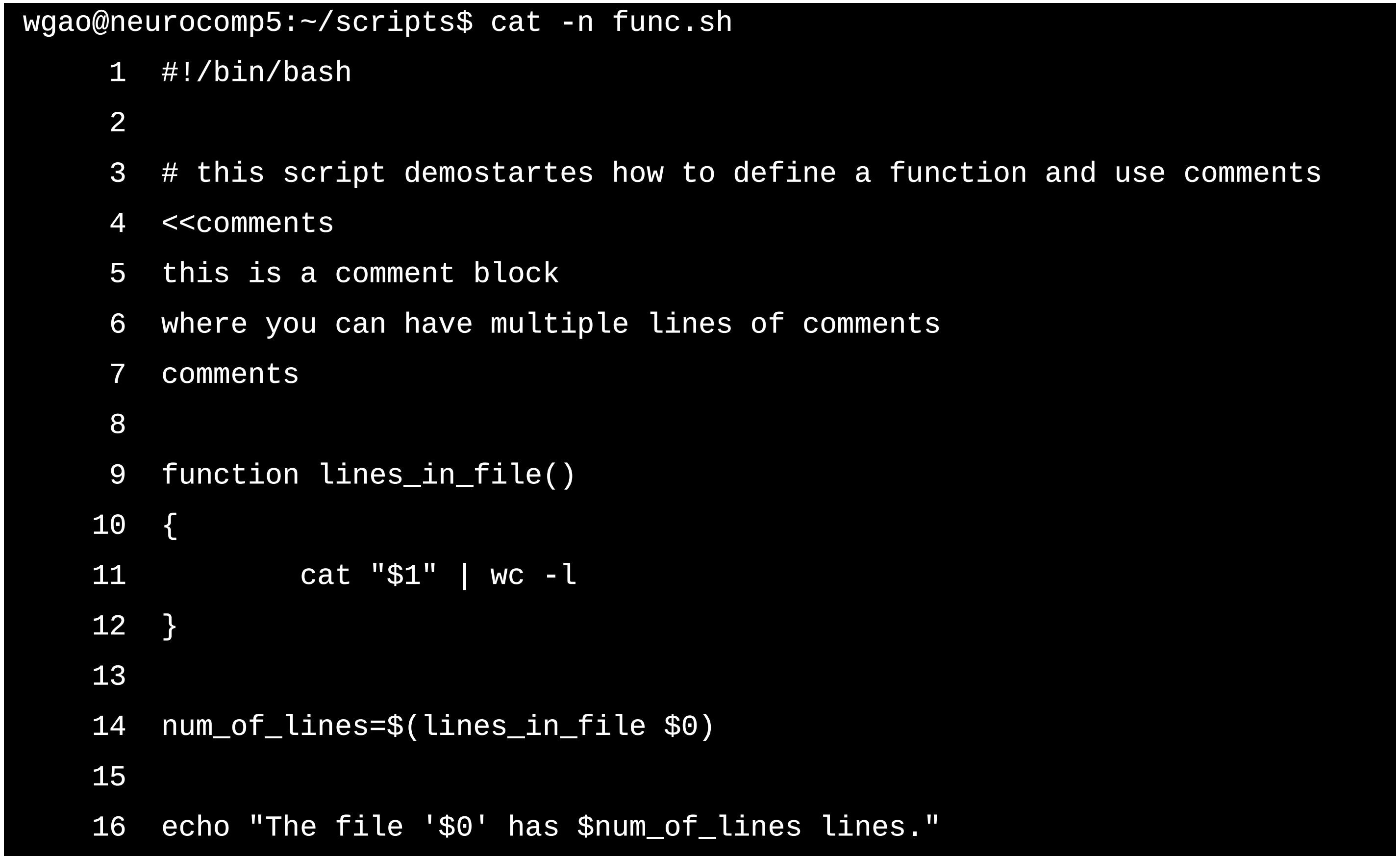
```
#!/bin/bash

read -p "Enter a numeric value (n1): " n1
read -p "Enter a numeric value (n2): " n2
if (($n1 == $n2)); then
        echo "n1 equals n2"
elif (($n1 > $n2)); then
        echo "n1 is greater than n2"
else
        echo "n1 is smaller than n2"
fi

read -p "Enter a string (s1): " s1
read -p "Enter a string (s2): " s2
if [ "$s1" == "$s2" ]; then
        echo "s1 is the same as s2"
elif [ "$s1" \< "$s2" ]; then
        echo "s1 is lexically smaller than s2"
else
        echo "s1 is lexically bigger than s2"
fi
```



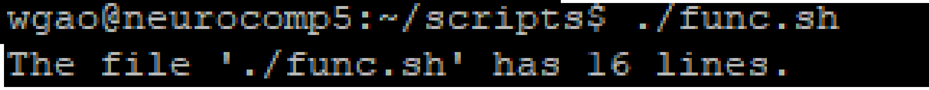
```
wgao@neurocomp5:~/scripts$ ./comparison.sh
Enter a numeric value (n1): 10
Enter a numeric value (n2): 2
n1 is greater than n2
Enter a string (s1): this
Enter a string (s2): that
s1 is lexically bigger than s2
```

BASH Script Examples

function



```
wgao@neurocomp5:~/scripts$ cat -n func.sh
     1  #!/bin/bash
     2
     3  # this script demostartes how to define a function and use comments
     4  <<comments
     5  this is a comment block
     6  where you can have multiple lines of comments
     7  comments
     8
     9  function lines_in_file()
    10  {
    11          cat "$1" | wc -l
    12  }
    13
    14  num_of_lines=$(lines_in_file $0)
    15
    16  echo "The file '$0' has $num_of_lines lines."
```

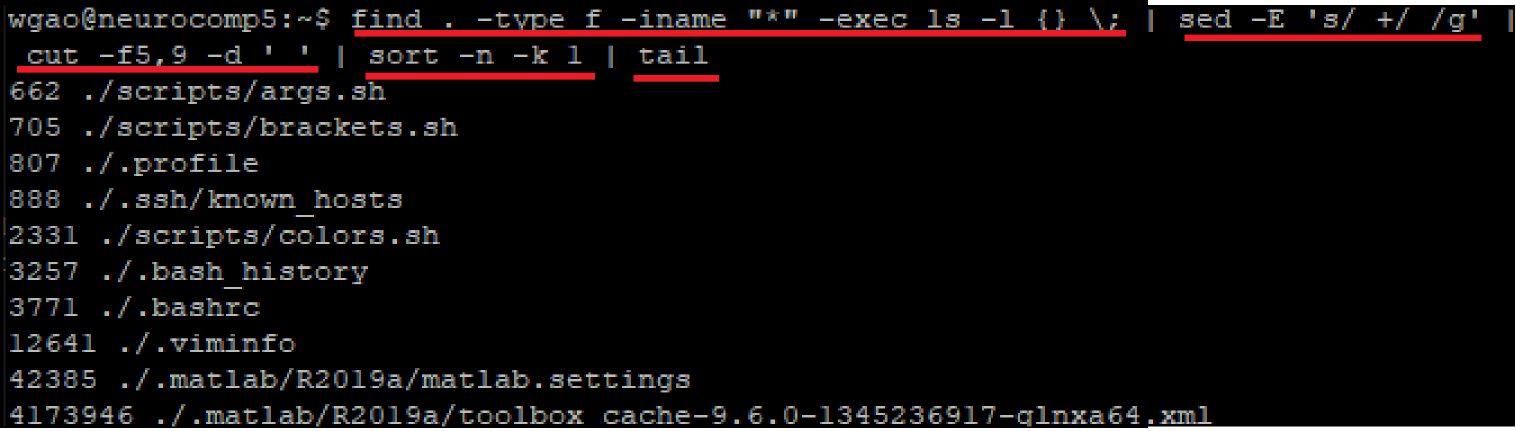


```
wgao@neurocomp5:~/scripts$ ./func.sh
The file './func.sh' has 16 lines.
```

BASH Script Examples

find, sed, cut, sort, tail, |

```
find . -type f -iname "*" -
exec ls -l {} \; | sed -E 's/
+/ /g' | cut -f5,9 -d ' ' |
sort -n -k 1
```



```
wgao@neurocomp5:~$ find . -type f -iname "*" -exec ls -l {} \; | sed -E 's/ +/ /g' |
 cut -f5,9 -d ' ' | sort -n -k 1 | tail
662 ./scripts/args.sh
705 ./scripts/brackets.sh
807 ./.profile
888 ./.ssh/known_hosts
2331 ./scripts/colors.sh
3257 ./.bash_history
3771 ./.bashrc
12641 ./.viminfo
42385 ./.matlab/R2019a/matlab.settings
4173946 ./.matlab/R2019a/toolbox cache-9.6.0-1345236917-glnxa64.xml
```

Basic Slurm Concepts

Resources

- Consumable Resources: Node, CPU, Memory, ~~GPU~~
- Partitions
  - A partition contains a group of nodes/resources
  - Each partition can be considered as a job queue with its own settings
- Node states (sinfo): idle, mix, alloc, drng, drain, down
- Resource Allocation Limits: amount of resources, time, ~~preemption~~, ~~accounting~~
- Software: consistent across a partition / partitions
- Data: user’s own data is accessible across the cluster

Basic Slurm Concepts

Job Scheduling

- Fairness rules can be implemented once they are needed

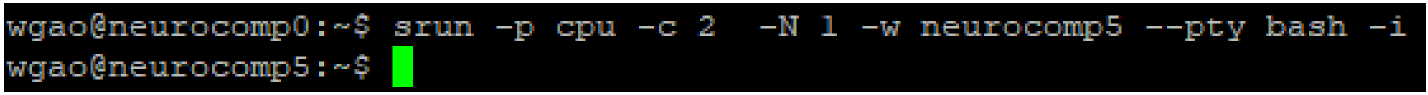
Basic Slurm Concepts

Job Termination

- Wall-time:
  - “cpu” partition: 30 days
  - “interactive” partition: 24 hours
  - Admin users can extend the wall-time of a running job
  - Wall-time is NOT CPU time
- scancel
  - *scancel JOBID*

Basic Slurm Commands

srun



```
wgao@neurocomp0:~$ srun -p cpu -c 2  -N 1 -w neurocomp5 --pty bash -i
wgao@neurocomp5:~$ 
```

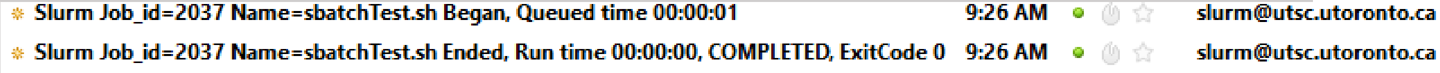
- -p: partition name
- -w: node name
- -c: number of CPU cores
- -N: number of nodes
- -mem: amount of memory
- --pty: interactive mode
- --x11: enable X forwarding (GUI)
- --mail-type: when to trigger a notification email (all)
- --mail-user: user's email address

Basic Slurm Commands

sbatch

```
#!/bin/bash
#SBATCH --job-name="sbatchTest.sh"
#SBATCH --partition=cpu
#SBATCH --nodes=1
#SBATCH --ntasks=12
#SBATCH --cpus-per-task=1
#SBATCH --mem=23000
#SBATCH --output=sbatchTest.sh.out
#SBATCH --mail-user=YOUR-EMAIL
#SBATCH --mail-type=ALL
##SBATCH --time=00:15:00
##SBATCH --requeue      #Specifies that the job will be requeued after a node failure. The default is that the job will not be requeued.
##SBATCH --checkpoint=1:0
hostname
```

```
wgao@neurocomp0:~$ sbatch –w neurocomp5 sbatchTest.sh
Submitted batch job 2037
wgao@neurocomp0:~$ cat sbatchTest.sh.out
neurocomp5
wgao@neurocomp0:~$
```



```
Slurm Job_id=2037 Name=sbatchTest.sh Began, Queued time 00:00:01               9:26 AM   slurm@utsc.utoronto.ca
Slurm Job_id=2037 Name=sbatchTest.sh Ended, Run time 00:00:00, COMPLETED, ExitCode 0   9:26 AM   slurm@utsc.utoronto.ca
```

Basic Slurm Commands

squeue

```
squeue -l -u USERNAME
squeue -l -t pending
squeue -l -t running
squeue -l -t all
```

-l, --long	long report
-t, --type	comma separated list of job states

…

Basic Slurm Commands

scontrol, scancel

```
scontrol show jobid <jobid>
scancel JOB-ID
```

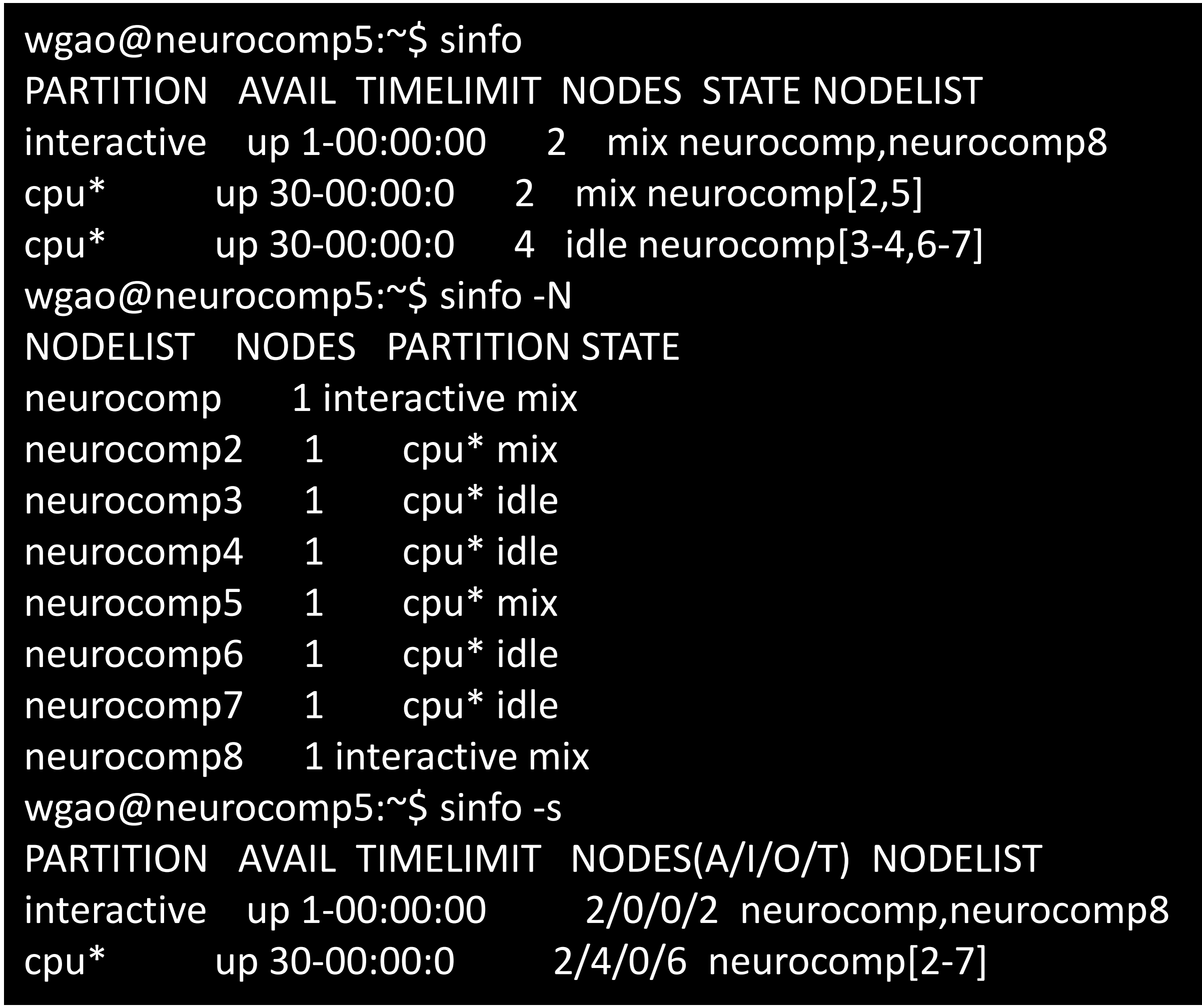
Basic Slurm Commands

sinfo

sinfo

sinfo -N

sinfo -s



```
wgao@neurocomp5:~$ sinfo
PARTITION   AVAIL  TIMELIMIT  NODES  STATE NODELIST
interactive    up 1-00:00:00      2    mix neurocomp,neurocomp8
cpu*           up 30-00:00:0      2    mix neurocomp[2,5]
cpu*           up 30-00:00:0      4   idle neurocomp[3-4,6-7]
wgao@neurocomp5:~$ sinfo -N
NODELIST    NODES   PARTITION STATE
neurocomp       1 interactive mix
neurocomp2      1        cpu* mix
neurocomp3      1        cpu* idle
neurocomp4      1        cpu* idle
neurocomp5      1        cpu* mix
neurocomp6      1        cpu* idle
neurocomp7      1        cpu* idle
neurocomp8      1 interactive mix
wgao@neurocomp5:~$ sinfo -s
PARTITION   AVAIL  TIMELIMIT   NODES(A/I/O/T)  NODELIST
interactive    up 1-00:00:00          2/0/0/2  neurocomp,neurocomp8
cpu*           up 30-00:00:0          2/4/0/6  neurocomp[2-7]
```

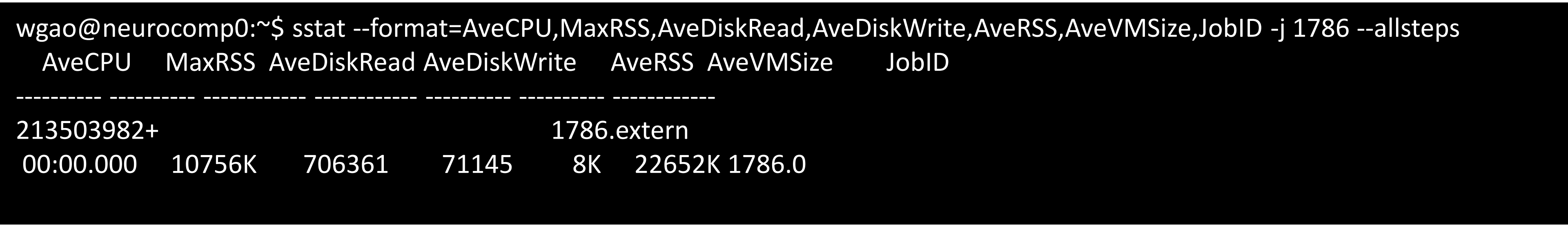
Basic Slurm Commands

sstat

sstat --helpformat

sstat --format=AveCPU,MaxRSS,AveDiskRead,AveDiskWrite,AveRSS,AveVMSize,JobID -j <JOBID> --allsteps

sstat -j <JOBID> --allsteps

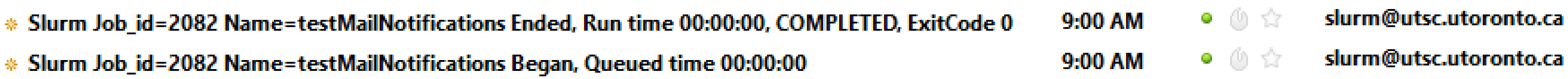


```
wgao@neurocomp0:~$ sstat --format=AveCPU,MaxRSS,AveDiskRead,AveDiskWrite,AveRSS,AveVMSize,JobID -j 1786 --allsteps
    AveCPU     MaxRSS  AveDiskRead AveDiskWrite     AveRSS  AveVMSize        JobID
---------- ---------- ------------ ------------ ---------- ---------- ------------
213503982+                                                 1786.extern
 00:00.000     10756K       706361        71145         8K     22652K 1786.0
```

Slurm Examples

srun – with E-mail notifications

srun -p interactive -N 1 -c 1 --job-name=testMailNotifications --mail-type=all --mail-user=YOUR-EMAIL hostname



Slurm Job_id=2082 Name=testMailNotifications Ended, Run time 00:00:00, COMPLETED, ExitCode 0 9:00 AM ● ☆ slurm@utsc.utoronto.ca

Slurm Job_id=2082 Name=testMailNotifications Began, Queued time 00:00:00 9:00 AM ● ☆ slurm@utsc.utoronto.ca

Slurm Examples

start a GUI Slurm session

- Linux
  - *ssh -X UTORid@neurocomp0.utsc.utoronto.ca*
- OS X
  - *ssh -X UTORid@neurocomp0.utsc.utoronto.ca*
  - *ssh -Y UTORid@neurocomp0.utsc.utoronto.ca*
- Windows / Linux / OS X
  - X2Go
    - Host: neurocomp0.utsc.utoronto.ca
    - Login: UTORid
    - Session type: XFCE

x2go

1) Connect to neurocomp0 using X2Go
2) Open a terminal
3) * Run "*ssh -X neurocomp0*"
4) Start a Slurm session using "srun" or "sbatch"

Slurm Examples

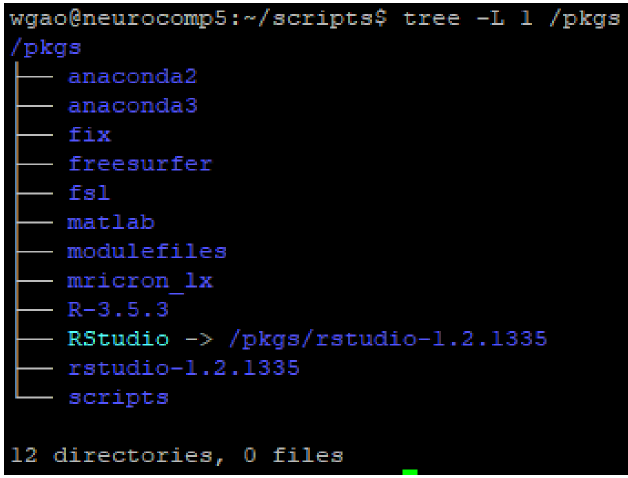
mpi

```
wgao@neurocomp0:~$ srun -p cpu -N 2 -c 2 --pty bash -i
wgao@neurocomp2:~$
wgao@neurocomp2:~$ mpirun hostname
neurocomp2
neurocomp3
wgao@neurocomp2:~$ squeue
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
              2083       cpu     bash     wgao  R       2:47      2 neurocomp[2-3]
```

More about Using Psy Cluster

/pkgs directory and *module* command

- Additional packages are available in /pkgs folder
- Psy cluster has four versions of Matlab available (*ls -l /opt/MATLAB*)
- module command can be used to load / unload environment modules
  - *module avail*
  - *module list*
  - *module load*
  - *module unload*
- Quota and Slurm stats @ neurocomp0
  - *quotaStats.sh*
  - *slurmStats.sh*



```
wgao@neurocomp5:~/scripts$ tree -L 1 /pkgs
/pkgs
├── anaconda2
├── anaconda3
├── fix
├── freesurfer
├── fsl
├── matlab
├── modulefiles
├── mricron_lx
├── R-3.5.3
├── RStudio -> /pkgs/rstudio-1.2.1335
├── rstudio-1.2.1335
└── scripts

12 directories, 0 files
```

More about Using Psy Cluster

- Cluster status and how-to pages
  - https://psycomp.utsc.utoronto.ca
- Quota monitoring
- Slurm session monitoring
- Debugging
- Code optimization

More Computing Support Topics…

- Checkpointing & Slurm + DMTCP
- Storage & Backup
- Linux Scripting
- Parallel Computing
- Best Practices & Optimization
- Debugging