CS2211b

# Software Tools and Systems Programming


## Week 4b

Shell Scripts: Part 1

# Announcements

Assignment #2 on OWL

# `top` Command

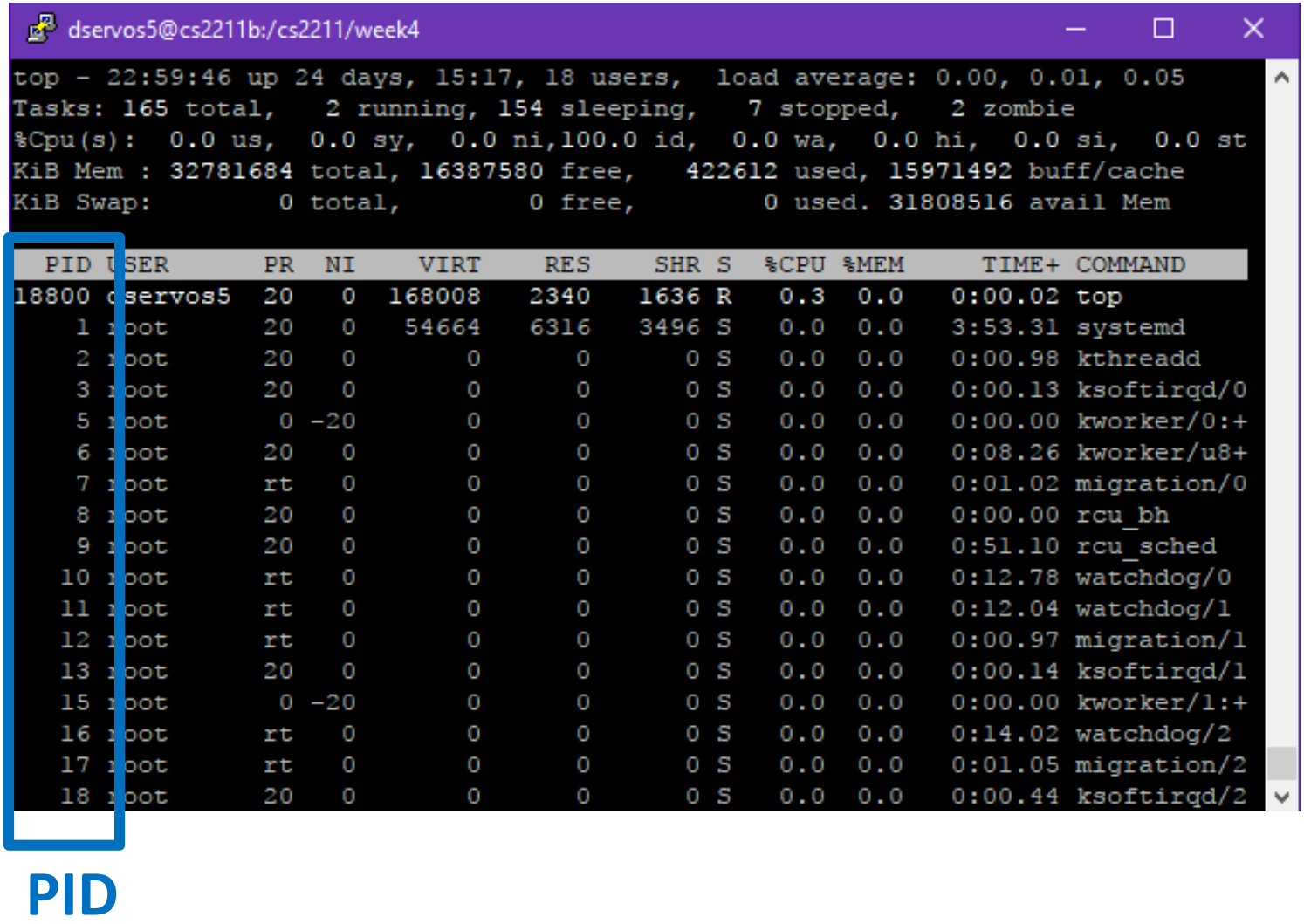
- **`top`** command displays the top *n* processes on the system by CPU time, memory, etc.
- Information is updated live to the terminal.
- Supports interactive commands while running:
  - h help
  - c display full command
  - M sort by memory usage
  - P sort by CPU usage (%)
  - T sort by time (CPU time)
  - i show/hide idle processes
  - n set the number of processes displayed
  - u display the processes of a given user
  - k kill a given process
  - q quit

# top Command

dservos5@cs2211b:/cs2211/week4

```
top - 22:59:46 up 24 days, 15:17, 18 users,  load average: 0.00, 0.01, 0.05
Tasks: 165 total,   2 running, 154 sleeping,   7 stopped,   2 zombie
%Cpu(s):  0.0 us,  0.0 sy,  0.0 ni,100.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
KiB Mem : 32781684 total, 16387580 free,   422612 used, 15971492 buff/cache
KiB Swap:        0 total,        0 free,        0 used. 31808516 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
18800 dservos5  20   0  168008   2340   1636 R   0.3  0.0   0:00.02 top
    1 root      20   0   54664   6316   3496 S   0.0  0.0   3:53.31 systemd
    2 root      20   0       0      0      0 S   0.0  0.0   0:00.98 kthreadd
    3 root      20   0       0      0      0 S   0.0  0.0   0:00.13 ksoftirqd/0
    5 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 kworker/0:+
    6 root      20   0       0      0      0 S   0.0  0.0   0:08.26 kworker/u8+
    7 root      rt   0       0      0      0 S   0.0  0.0   0:01.02 migration/0
    8 root      20   0       0      0      0 S   0.0  0.0   0:00.00 rcu_bh
    9 root      20   0       0      0      0 S   0.0  0.0   0:51.10 rcu_sched
   10 root      rt   0       0      0      0 S   0.0  0.0   0:12.78 watchdog/0
   11 root      rt   0       0      0      0 S   0.0  0.0   0:12.04 watchdog/1
   12 root      rt   0       0      0      0 S   0.0  0.0   0:00.97 migration/1
   13 root      20   0       0      0      0 S   0.0  0.0   0:00.14 ksoftirqd/1
   15 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 kworker/1:+
   16 root      rt   0       0      0      0 S   0.0  0.0   0:14.02 watchdog/2
   17 root      rt   0       0      0      0 S   0.0  0.0   0:01.05 migration/2
   18 root      20   0       0      0      0 S   0.0  0.0   0:00.44 ksoftirqd/2
```

# top Command




**PID**

# top Command

dservos5@cs2211b:/cs2211/week4

```
top - 22:59:46 up 24 days, 15:17, 18 users,  load average: 0.00, 0.01, 0.05
Tasks: 165 total,   2 running, 154 sleeping,   7 stopped,   2 zombie
%Cpu(s):  0.0 us,  0.0 sy,  0.0 ni,100.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
KiB Mem : 32781684 total, 16387580 free,   422612 used, 15971492 buff/cache
KiB Swap:        0 total,        0 free,        0 used. 31808516 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
18800 dservos5  20   0  168008   2340   1636 R   0.3  0.0   0:00.02 top
    1 root      20   0   54664   6316   3496 S   0.0  0.0   3:53.31 systemd
    2 root      20   0       0      0      0 S   0.0  0.0   0:00.98 kthreadd
    3 root      20   0       0      0      0 S   0.0  0.0   0:00.13 ksoftirqd/0
    5 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 kworker/0:+
    6 root      20   0       0      0      0 S   0.0  0.0   0:08.26 kworker/u8+
    7 root      rt   0       0      0      0 S   0.0  0.0   0:01.02 migration/0
    8 root      20   0       0      0      0 S   0.0  0.0   0:00.00 rcu_bh
    9 root      20   0       0      0      0 S   0.0  0.0   0:51.10 rcu_sched
   10 root      rt   0       0      0      0 S   0.0  0.0   0:12.78 watchdog/0
   11 root      rt   0       0      0      0 S   0.0  0.0   0:12.04 watchdog/1
   12 root      rt   0       0      0      0 S   0.0  0.0   0:00.97 migration/1
   13 root      20   0       0      0      0 S   0.0  0.0   0:00.14 ksoftirqd/1
   15 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 kworker/1:+
   16 root      rt   0       0      0      0 S   0.0  0.0   0:14.02 watchdog/2
   17 root      rt   0       0      0      0 S   0.0  0.0   0:01.05 migration/2
   18 root      20   0       0      0      0 S   0.0  0.0   0:00.44 ksoftirqd/2
```

**User Name**

# top Command

dservos5@cs2211b:/cs2211/week4

```
top - 22:59:46 up 24 days, 15:17, 18 users,  load average: 0.00, 0.01, 0.05
Tasks: 165 total,   2 running, 154 sleeping,   7 stopped,   2 zombie
%Cpu(s):  0.0 us,  0.0 sy,  0.0 ni,100.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
KiB Mem : 32781684 total, 16387580 free,   422612 used, 15971492 buff/cache
KiB Swap:        0 total,        0 free,        0 used. 31808516 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
18800 dservos5  20   0  168008   2340   1636 R   0.3  0.0   0:00.02 top
    1 root      20   0   54664   6316   3496 S   0.0  0.0   3:53.31 systemd
    2 root      20   0       0      0      0 S   0.0  0.0   0:00.98 kthreadd
    3 root      20   0       0      0      0 S   0.0  0.0   0:00.13 ksoftirqd/0
    5 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 kworker/0:+
    6 root      20   0       0      0      0 S   0.0  0.0   0:08.26 kworker/u8+
    7 root      rt   0       0      0      0 S   0.0  0.0   0:01.02 migration/0
    8 root      20   0       0      0      0 S   0.0  0.0   0:00.00 rcu_bh
    9 root      20   0       0      0      0 S   0.0  0.0   0:51.10 rcu_sched
   10 root      rt   0       0      0      0 S   0.0  0.0   0:12.78 watchdog/0
   11 root      rt   0       0      0      0 S   0.0  0.0   0:12.04 watchdog/1
   12 root      rt   0       0      0      0 S   0.0  0.0   0:00.97 migration/1
   13 root      20   0       0      0      0 S   0.0  0.0   0:00.14 ksoftirqd/1
   15 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 kworker/1:+
   16 root      rt   0       0      0      0 S   0.0  0.0   0:14.02 watchdog/2
   17 root      rt   0       0      0      0 S   0.0  0.0   0:01.05 migration/2
   18 root      20   0       0      0      0 S   0.0  0.0   0:00.44 ksoftirqd/2
```

**Priority**

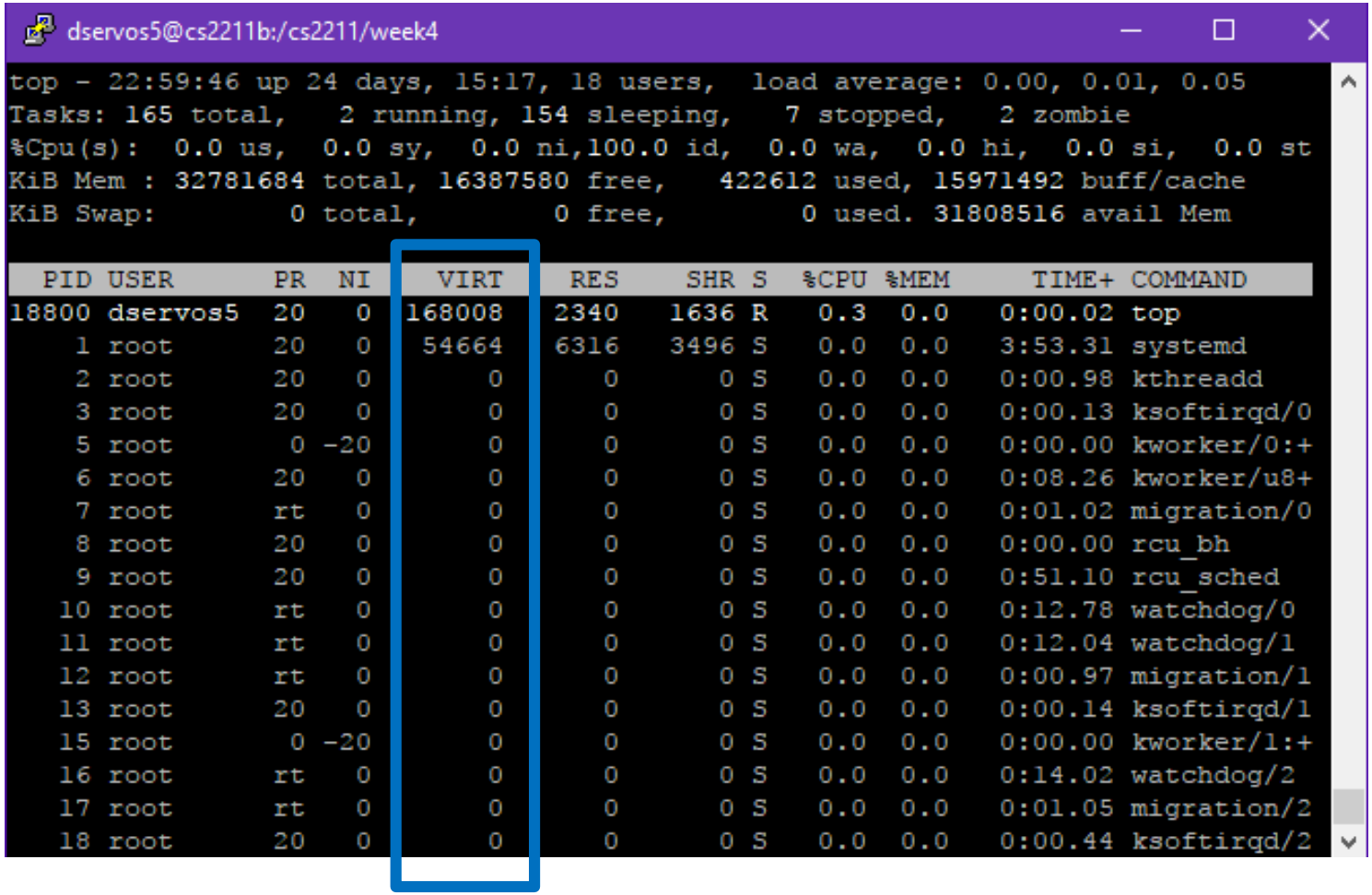
# top Command

dservos5@cs2211b:/cs2211/week4

```
top - 22:59:46 up 24 days, 15:17, 18 users,  load average: 0.00, 0.01, 0.05
Tasks: 165 total,   2 running, 154 sleeping,   7 stopped,   2 zombie
%Cpu(s):  0.0 us,  0.0 sy,  0.0 ni,100.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
KiB Mem : 32781684 total, 16387580 free,   422612 used, 15971492 buff/cache
KiB Swap:        0 total,        0 free,        0 used. 31808516 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
18800 dservos5  20   0  168008   2340   1636 R   0.3  0.0   0:00.02 top
    1 root      20   0   54664   6316   3496 S   0.0  0.0   3:53.31 systemd
    2 root      20   0       0      0      0 S   0.0  0.0   0:00.98 kthreadd
    3 root      20   0       0      0      0 S   0.0  0.0   0:00.13 ksoftirqd/0
    5 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 kworker/0:+
    6 root      20   0       0      0      0 S   0.0  0.0   0:08.26 kworker/u8+
    7 root      rt   0       0      0      0 S   0.0  0.0   0:01.02 migration/0
    8 root      20   0       0      0      0 S   0.0  0.0   0:00.00 rcu_bh
    9 root      20   0       0      0      0 S   0.0  0.0   0:51.10 rcu_sched
   10 root      rt   0       0      0      0 S   0.0  0.0   0:12.78 watchdog/0
   11 root      rt   0       0      0      0 S   0.0  0.0   0:12.04 watchdog/1
   12 root      rt   0       0      0      0 S   0.0  0.0   0:00.97 migration/1
   13 root      20   0       0      0      0 S   0.0  0.0   0:00.14 ksoftirqd/1
   15 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 kworker/1:+
   16 root      rt   0       0      0      0 S   0.0  0.0   0:14.02 watchdog/2
   17 root      rt   0       0      0      0 S   0.0  0.0   0:01.05 migration/2
   18 root      20   0       0      0      0 S   0.0  0.0   0:00.44 ksoftirqd/2
```

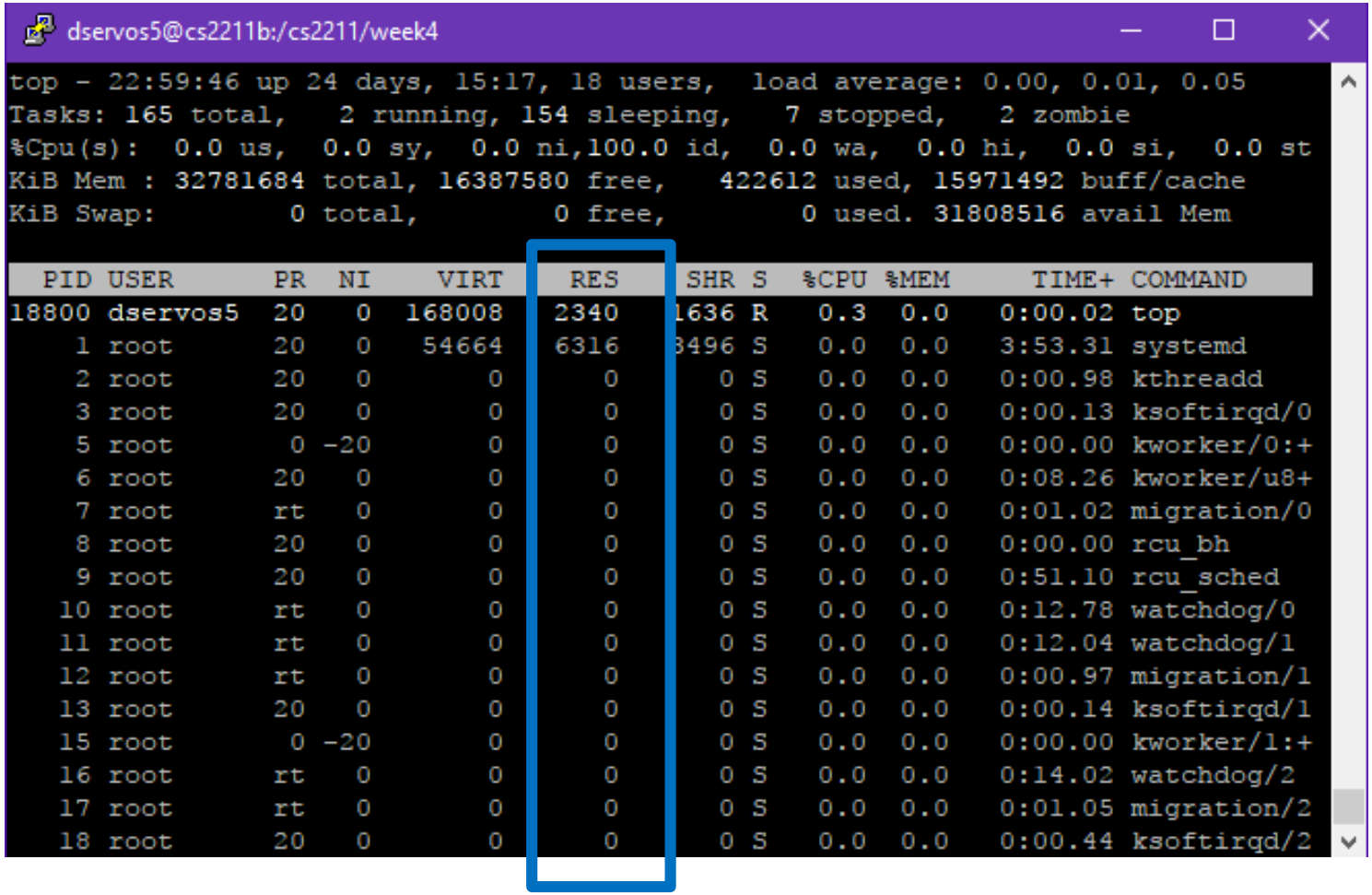
**Nice Value**

# top Command



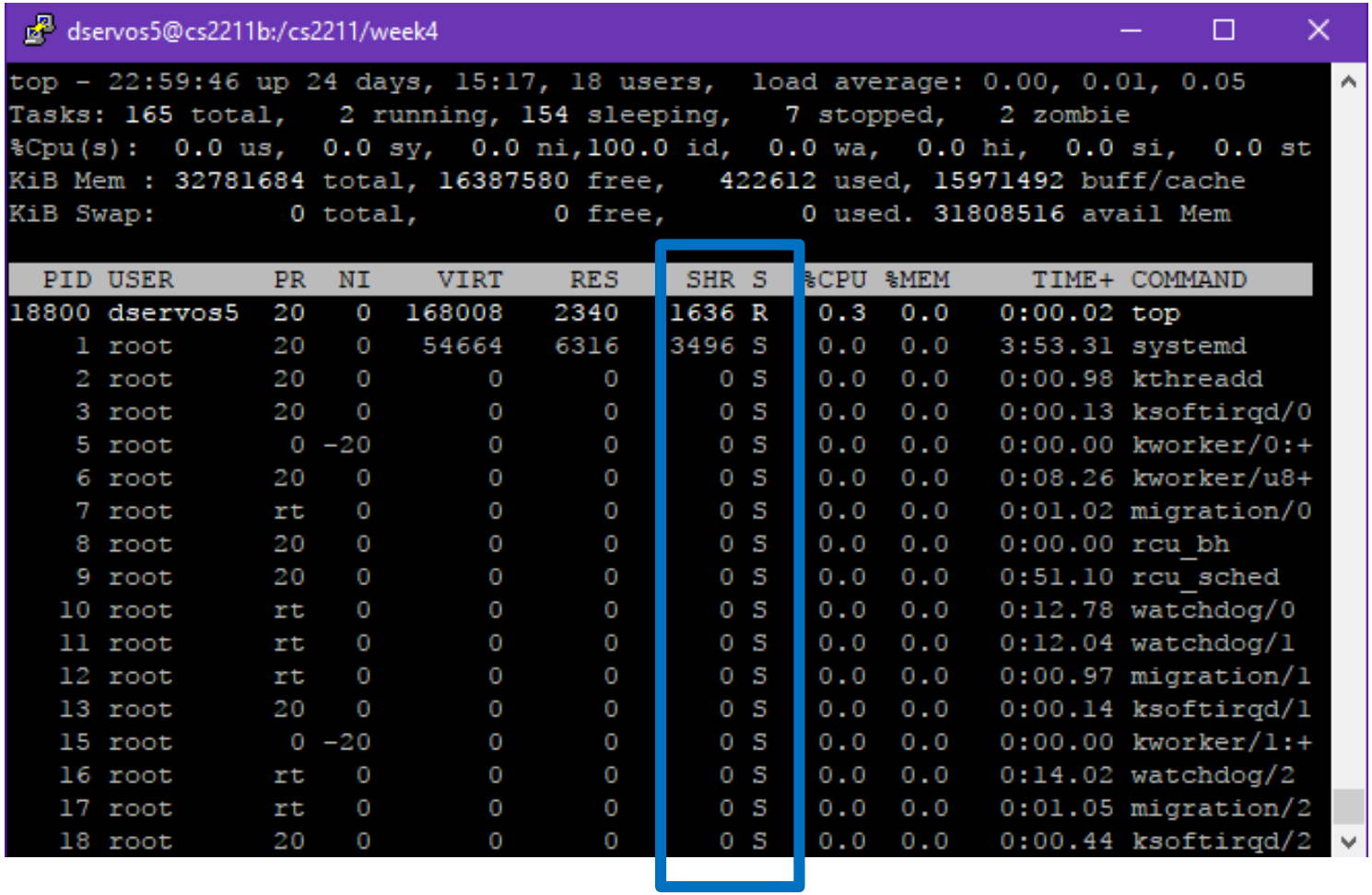

**Virtual Memory Size**

# top Command




**Resident Size (Physical Memory)**

# top Command




**How much of the VIRT size is shared/shareable**

# top Command

dservos5@cs2211b:/cs2211/week4

```
top - 22:59:46 up 24 days, 15:17, 18 users,  load average: 0.00, 0.01, 0.05
Tasks: 165 total,   2 running, 154 sleeping,   7 stopped,   2 zombie
%Cpu(s):  0.0 us,  0.0 sy,  0.0 ni,100.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
KiB Mem : 32781684 total, 16387580 free,   422612 used, 15971492 buff/cache
KiB Swap:        0 total,        0 free,        0 used. 31808516 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
18800 dservos5  20   0  168008   2340   1636 R   0.3  0.0   0:00.02 top
    1 root      20   0   54664   6316   3496 S   0.0  0.0   3:53.31 systemd
    2 root      20   0       0      0      0 S   0.0  0.0   0:00.98 kthreadd
    3 root      20   0       0      0      0 S   0.0  0.0   0:00.13 ksoftirqd/0
    5 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 kworker/0:+
    6 root      20   0       0      0      0 S   0.0  0.0   0:08.26 kworker/u8+
    7 root      rt   0       0      0      0 S   0.0  0.0   0:01.02 migration/0
    8 root      20   0       0      0      0 S   0.0  0.0   0:00.00 rcu_bh
    9 root      20   0       0      0      0 S   0.0  0.0   0:51.10 rcu_sched
   10 root      rt   0       0      0      0 S   0.0  0.0   0:12.78 watchdog/0
   11 root      rt   0       0      0      0 S   0.0  0.0   0:12.04 watchdog/1
   12 root      rt   0       0      0      0 S   0.0  0.0   0:00.97 migration/1
   13 root      20   0       0      0      0 S   0.0  0.0   0:00.14 ksoftirqd/1
   15 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 kworker/1:+
   16 root      rt   0       0      0      0 S   0.0  0.0   0:14.02 watchdog/2
   17 root      rt   0       0      0      0 S   0.0  0.0   0:01.05 migration/2
   18 root      20   0       0      0      0 S   0.0  0.0   0:00.44 ksoftirqd/2
```

**CPU Usage**

# top Command

dservos5@cs2211b:/cs2211/week4

```
top - 22:59:46 up 24 days, 15:17, 18 users,  load average: 0.00, 0.01, 0.05
Tasks: 165 total,   2 running, 154 sleeping,   7 stopped,   2 zombie
%Cpu(s):  0.0 us,  0.0 sy,  0.0 ni,100.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
KiB Mem : 32781684 total, 16387580 free,   422612 used, 15971492 buff/cache
KiB Swap:        0 total,        0 free,        0 used. 31808516 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
18800 dservos5  20   0  168008   2340   1636 R   0.3  0.0   0:00.02 top
    1 root      20   0   54664   6316   3496 S   0.0  0.0   3:53.31 systemd
    2 root      20   0       0      0      0 S   0.0  0.0   0:00.98 kthreadd
    3 root      20   0       0      0      0 S   0.0  0.0   0:00.13 ksoftirqd/0
    5 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 kworker/0:+
    6 root      20   0       0      0      0 S   0.0  0.0   0:08.26 kworker/u8+
    7 root      rt   0       0      0      0 S   0.0  0.0   0:01.02 migration/0
    8 root      20   0       0      0      0 S   0.0  0.0   0:00.00 rcu_bh
    9 root      20   0       0      0      0 S   0.0  0.0   0:51.10 rcu_sched
   10 root      rt   0       0      0      0 S   0.0  0.0   0:12.78 watchdog/0
   11 root      rt   0       0      0      0 S   0.0  0.0   0:12.04 watchdog/1
   12 root      rt   0       0      0      0 S   0.0  0.0   0:00.97 migration/1
   13 root      20   0       0      0      0 S   0.0  0.0   0:00.14 ksoftirqd/1
   15 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 kworker/1:+
   16 root      rt   0       0      0      0 S   0.0  0.0   0:14.02 watchdog/2
   17 root      rt   0       0      0      0 S   0.0  0.0   0:01.05 migration/2
   18 root      20   0       0      0      0 S   0.0  0.0   0:00.44 ksoftirqd/2
```

**Memory Usage (Physical Only)**

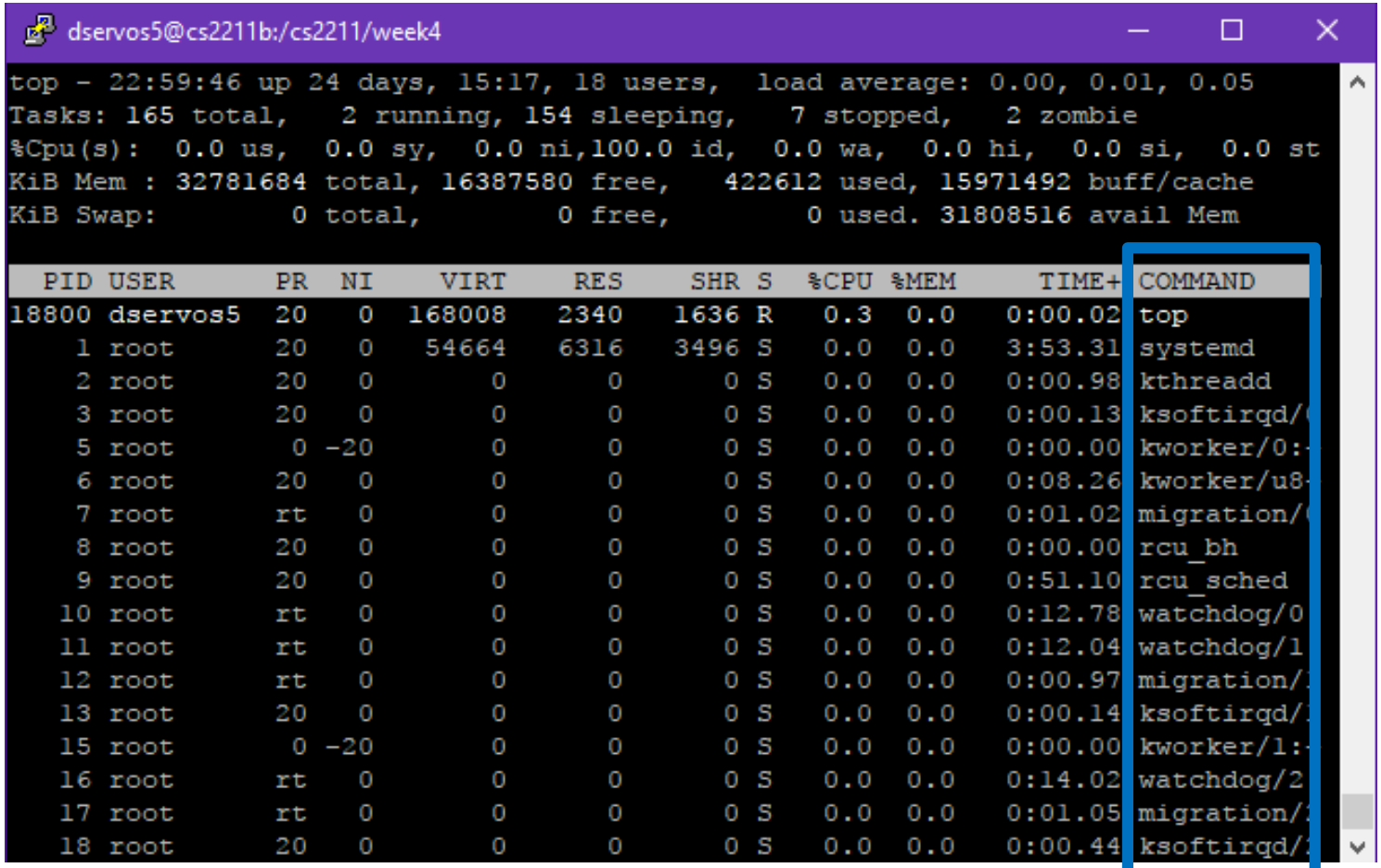
# top Command

dservos5@cs2211b:/cs2211/week4

```
top - 22:59:46 up 24 days, 15:17, 18 users,  load average: 0.00, 0.01, 0.05
Tasks: 165 total,   2 running, 154 sleeping,   7 stopped,   2 zombie
%Cpu(s):  0.0 us,  0.0 sy,  0.0 ni,100.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
KiB Mem : 32781684 total, 16387580 free,   422612 used, 15971492 buff/cache
KiB Swap:        0 total,        0 free,        0 used. 31808516 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
18800 dservos5  20   0  168008   2340   1636 R   0.3  0.0   0:00.02 top
    1 root      20   0   54664   6316   3496 S   0.0  0.0   3:53.31 systemd
    2 root      20   0       0      0      0 S   0.0  0.0   0:00.98 kthreadd
    3 root      20   0       0      0      0 S   0.0  0.0   0:00.13 ksoftirqd/0
    5 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 kworker/0:+
    6 root      20   0       0      0      0 S   0.0  0.0   0:08.26 kworker/u8+
    7 root      rt   0       0      0      0 S   0.0  0.0   0:01.02 migration/0
    8 root      20   0       0      0      0 S   0.0  0.0   0:00.00 rcu_bh
    9 root      20   0       0      0      0 S   0.0  0.0   0:51.10 rcu_sched
   10 root      rt   0       0      0      0 S   0.0  0.0   0:12.78 watchdog/0
   11 root      rt   0       0      0      0 S   0.0  0.0   0:12.04 watchdog/1
   12 root      rt   0       0      0      0 S   0.0  0.0   0:00.97 migration/1
   13 root      20   0       0      0      0 S   0.0  0.0   0:00.14 ksoftirqd/1
   15 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 kworker/1:+
   16 root      rt   0       0      0      0 S   0.0  0.0   0:14.02 watchdog/2
   17 root      rt   0       0      0      0 S   0.0  0.0   0:01.05 migration/2
   18 root      20   0       0      0      0 S   0.0  0.0   0:00.44 ksoftirqd/2
```

**CPU Time**

# top Command


dservos5@cs2211b:/cs2211/week4
top - 22:59:46 up 24 days, 15:17, 18 users, load average: 0.00, 0.01, 0.05
Tasks: 165 total, 2 running, 154 sleeping, 7 stopped, 2 zombie
%Cpu(s): 0.0 us, 0.0 sy, 0.0 ni,100.0 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
KiB Mem : 32781684 total, 16387580 free, 422612 used, 15971492 buff/cache
KiB Swap: 0 total, 0 free, 0 used. 31808516 avail Mem
PID USER PR NI VIRT RES SHR S %CPU %MEM TIME+ COMMAND
18800 dservos5 20 0 168008 2340 1636 R 0.3 0.0 0:00.02 top
1 root 20 0 54664 6316 3496 S 0.0 0.0 3:53.31 systemd
2 root 20 0 0 0 0 S 0.0 0.0 0:00.98 kthreadd
3 root 20 0 0 0 0 S 0.0 0.0 0:00.13 ksoftirqd/
5 root 0 -20 0 0 0 S 0.0 0.0 0:00.00 kworker/0:
6 root 20 0 0 0 0 S 0.0 0.0 0:08.26 kworker/u8
7 root rt 0 0 0 0 S 0.0 0.0 0:01.02 migration/
8 root 20 0 0 0 0 S 0.0 0.0 0:00.00 rcu_bh
9 root 20 0 0 0 0 S 0.0 0.0 0:51.10 rcu_sched
10 root rt 0 0 0 0 S 0.0 0.0 0:12.78 watchdog/0
11 root rt 0 0 0 0 S 0.0 0.0 0:12.04 watchdog/1
12 root rt 0 0 0 0 S 0.0 0.0 0:00.97 migration/
13 root 20 0 0 0 0 S 0.0 0.0 0:00.14 ksoftirqd/
15 root 0 -20 0 0 0 S 0.0 0.0 0:00.00 kworker/1:
16 root rt 0 0 0 0 S 0.0 0.0 0:14.02 watchdog/2
17 root rt 0 0 0 0 S 0.0 0.0 0:01.05 migration/
18 root 20 0 0 0 0 S 0.0 0.0 0:00.44 ksoftirqd/
Command

# Shell Variables

# Shell Variables

- We have already seen a few shell variables:
  - `$PATH`
  - `$HOME`
  - `$SHELL`
- Similar to variables in programming languages.
- Store either a string of characters or a *NULL* value.
- Variable names must follow this pattern:

```
[a-zA-Z_][a-zA-Z0-9_]*
```

- Shell variable names are case sensitive.

# Shell Variables

**Which of the following are valid shell variable names?**

```
__TEST__
1ST_NAME
First_Name
My1stName
100%
OneHundred%
One100Percent
CAT+DOG
alllowercase
```

# Shell Variables

- We can access a shell variables value using the $ character in front of the variable name.
- **Examples:**

```
[dservos5@cs2211b ~]$ echo $PATH
/usr/local/bin:/usr/bin:/usr/local/sbin:/usr/sbin:/cs2211/bin:.:/gaul/s1/student/1985/dservos5/.local/bin:/gaul/s1/student/1985/dservos5/bin
```

# Shell Variables

- We can access a shell variables value using the $ character in front of the variable name.
- **Examples:**

```
[dservos5@cs2211b bin]$ cd $HOME
[dservos5@cs2211b ~]$
```

# Shell Variables

- We can access a shell variables value using the $ character in front of the variable name.
- **Examples:**

```
[dservos5@cs2211b ~]$ ls -l $SHELL
-rwxr-xr-x. 1 root root 960608 Sep  6 12:25 /bin/bash
```

# Shell Variables

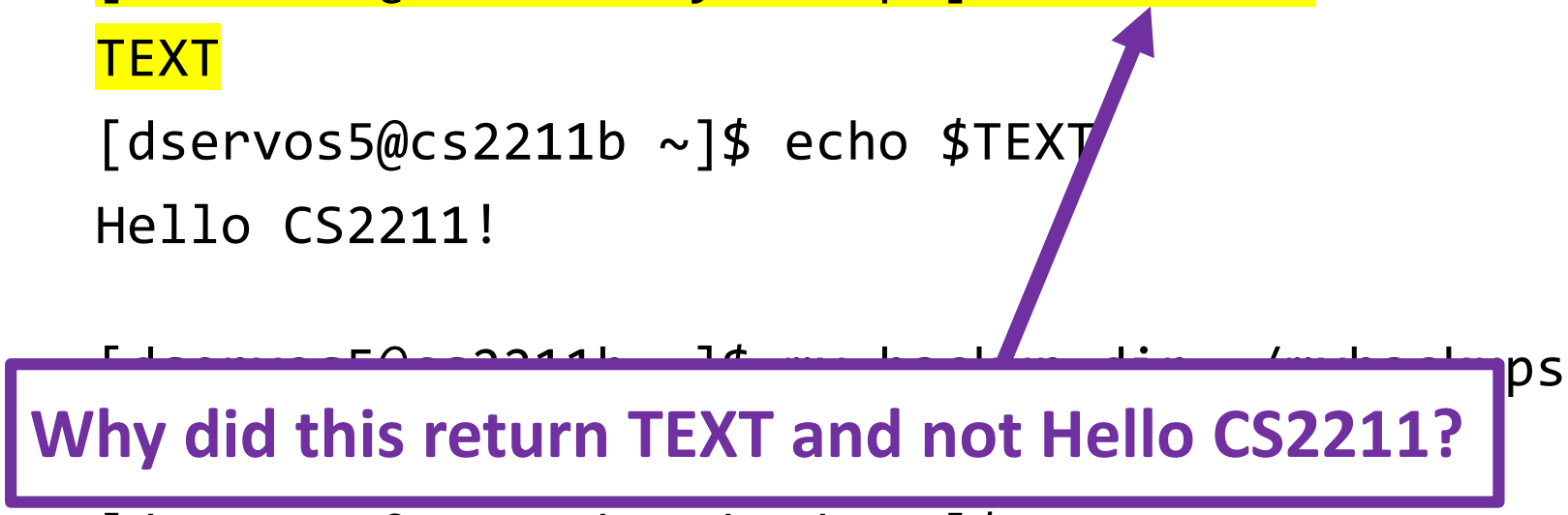
- We can set the value of a shell variable using the = operator.
- **Examples:**

```
[dservos5@cs2211b ~]$ TEXT='Hello CS2211!’
[dservos5@cs2211b mybackups]$ echo TEXT
TEXT
[dservos5@cs2211b ~]$ echo $TEXT
Hello CS2211!

[dservos5@cs2211b ~]$ my_backup_dir=~/mybackups
[dservos5@cs2211b ~]$ cd $my_backup_dir
[dservos5@cs2211b mybackups]$
```

# Shell Variables

- We can set the value of a shell variable using the = operator.
- **Examples:**

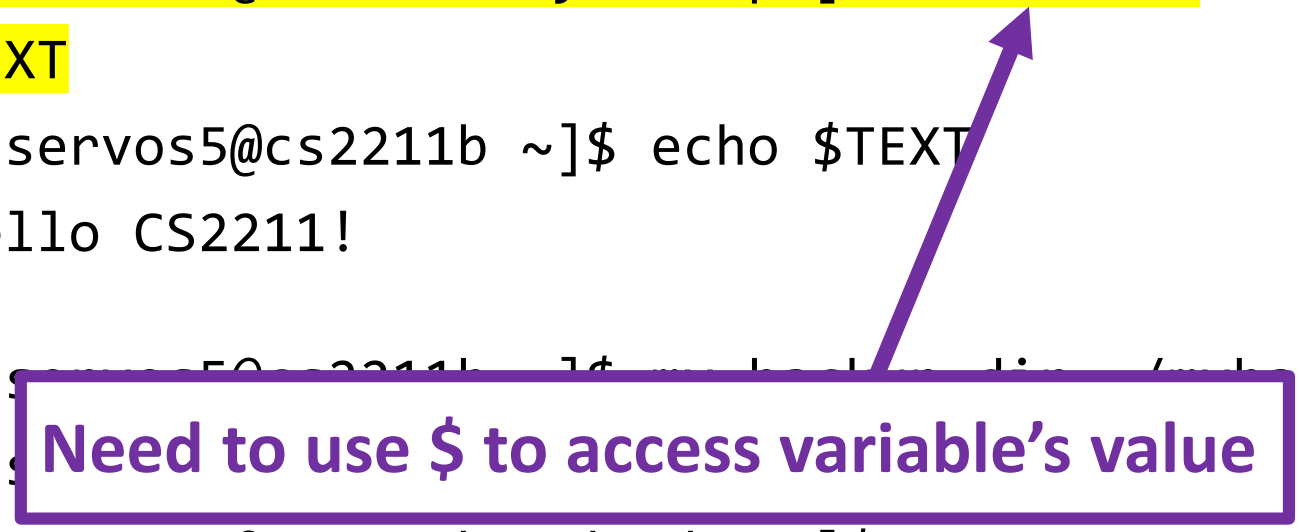
```
[dservos5@cs2211b ~]$ TEXT='Hello CS2211!'
[dservos5@cs2211b mybackups]$ echo TEXT
TEXT
[dservos5@cs2211b ~]$ echo $TEXT
Hello CS2211!

[dservos5@cs2211b ~]$ mybackup.sh /mybackups
[dservos5@cs2211b mybackups]$
```



# Shell Variables

- We can set the value of a shell variable using the = operator.
- **Examples:**

```
[dservos5@cs2211b ~]$ TEXT='Hello CS2211!'
[dservos5@cs2211b mybackups]$ echo TEXT
TEXT
[dservos5@cs2211b ~]$ echo $TEXT
Hello CS2211!

[ds                                    ckups
[ds
[dservos5@cs2211b mybackups]$
```



# Shell Variables

- We can set the value of a shell variable using the = operator.
- **Examples:**

```
[dservos5@cs2211b ~]$ TEXT='Hello CS2211!’
[dservos5@cs2211b mybackups]$ echo TEXT
TEXT
[dservos5@cs2211b ~]$ echo $TEXT
Hello CS2211!

[dservos5@cs2211b ~]$ my_backup_dir=~/mybackups
[dservos5@cs2211b ~]$ cd $my_backup_dir
[dservos5@cs2211b mybackups]$
```

# Shell Variables

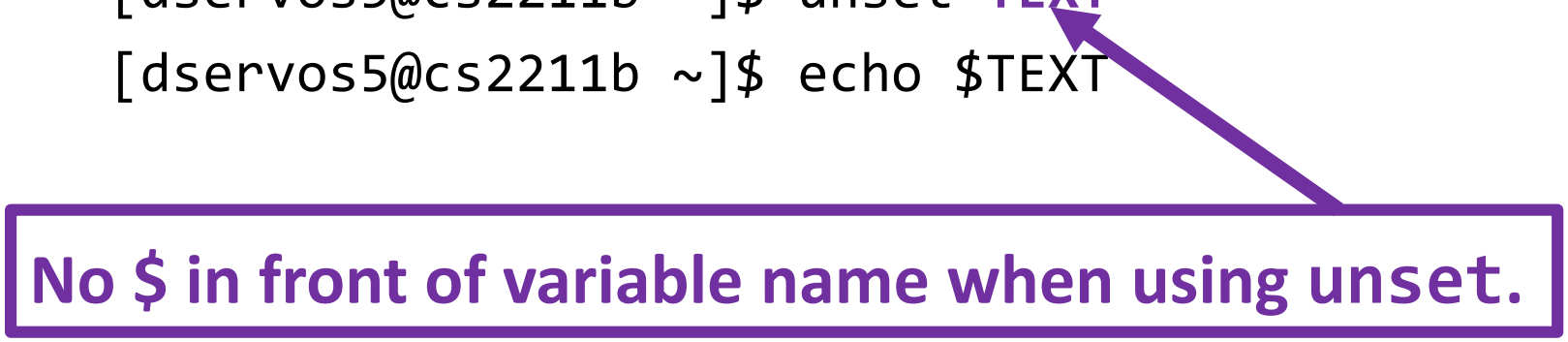
- We can unset or remove are variable using the **`unset`** command.
- **Examples:**

```
[dservos5@cs2211b ~]$ TEXT='Hello CS2211!’
[dservos5@cs2211b ~]$ echo $TEXT
Hello CS2211!
[dservos5@cs2211b ~]$ unset TEXT
[dservos5@cs2211b ~]$ echo $TEXT

[dservos5@cs2211b ~]$
```

# Shell Variables

- We can unset or remove are variable using the **`unset`** command.
- **Examples:**

```
[dservos5@cs2211b ~]$ TEXT='Hello CS2211!’
[dservos5@cs2211b ~]$ echo $TEXT
Hello CS2211!
[dservos5@cs2211b ~]$ unset TEXT
[dservos5@cs2211b ~]$ echo $TEXT
```



# Shell Variables

There two kinds of shell variables:

- **Environment** Variables (GLOBAL Variables)
  - Describes and affects the shell environment and programs invoked by the shell.
  - Inherited by child processes (including shell scripts)
  - Often created and set by the shell when initialized.

- **Regular** Variables (LOCAL Variables)
  - Affect the current shell but not programs invoked by the shell.
  - Not normally inherited by child processes (including shell scripts)
  - Often user defined.

# Shell Variables

## Example:

**/cs2211/week4/printvar**

```
#!/bin/bash

echo -n '$HOME = '
echo $HOME
echo -n '$SHELL = '
echo $SHELL
echo -n '$MYVAR = '
echo $MYVAR
```

**Output:**

```
[dservos5@cs2211b ~]$ /cs2211/week4/printvar
$HOME = /gaul/s1/student/1985/dservos5
$SHELL = /bin/bash
$MYVAR =
```

# Shell Variables

## Example:

**/cs2211/week4/printvar**

```
#!/bin/bash

echo -n '$HOME = '
echo $HOME
echo -n '$SHELL = '
echo $SHELL
echo -n '$MYVAR = '
echo $M
```

**$MYVAR has no value, not set yet.**

**Output:**

```
[dservos5@cs2211b ~]$ /cs2211/week4/printvar
$HOME = /gaul/s1/student/1985/dservos5
$SHELL = /bin/bash
$MYVAR =
```

# Shell Variables

## Example:

**Set value of $MYVAR and change value of $SHELL:**

```
[dservos5@cs2211b ~]$ MYVAR='This is the value of MYVAR.'
[dservos5@cs2211b ~]$ echo $MYVAR
This is the value of MYVAR.
[dservos5@cs2211b ~]$ SHELL='/bin/sh'
[dservos5@cs2211b ~]$ echo $SHELL
/bin/sh
[dservos5@cs2211b ~]$ /cs2211/week4/printvar
$HOME = /gaul/s1/student/1985/dservos5
$SHELL = /bin/sh
$MYVAR =
```

# Shell Variables

## Example:

**Set value of $MYVAR and change value of $SHELL:**

```
[dservos5@cs2211b ~]$ MYVAR='This is the value of MYVAR.'
[dservos5@cs2211b ~]$ echo $MYVAR
This is the value of MYVAR.
[dservos5@cs2211b ~]$ SHELL='/bin/sh'
[dservos5@cs2211b ~]$ echo $SHELL
/bin/sh
[dservos5@cs2211b ~]$ /cs2211/week4/printvar
$HOME = /gaul/s1/student/1985/dservos5
$SHELL = /bin/sh
$MYVAR =
```

**$SHELL was updated in *printvar* but not $MYVAR**

# Shell Variables

**$SHELL is an environment variable (global variable) that is inherited by child processes or scripts.**

**$MYVAR is a regular variable (local variable) that is local to the shell and not inherited by child processes or scripts.**

```
[dservos5@cs2211b ~]$ /cs2211/week4/printvar
$HOME = /gaul/s1/student/1985/dservos5
$SHELL = /bin/sh
$MYVAR =
```

# Shell Variables

- How can we make $MYVAR an environment variable such that it will be output by *printvar*?
- We can use the **`export`** command to make a variable available to all child processes and scripts.
- **Example:**

```
[dservos5@cs2211b ~]$ MYVAR='Lets make this global!'
[dservos5@cs2211b ~]$ export MYVAR
[dservos5@cs2211b ~]$ /cs2211/week4/printvar
$HOME = /gaul/s1/student/1985/dservos5
$SHELL = /bin/sh
$MYVAR = Lets make this global!
```

# Shell Variables

- How can we make $MYVAR an environment variable such that it will be output by *printvar*?
- We can use the **export** command to make a variable
- **Example:**

**No $ in front of variable name when using export.**

```
[dservos5@cs2211b ~]$ MYVAR='Lets make this global!'
[dservos5@cs2211b ~]$ export MYVAR
[dservos5@cs2211b ~]$ /cs2211/week4/printvar
$HOME = /gaul/s1/student/1985/dservos5
$SHELL = /bin/sh
$MYVAR = Lets make this global!
```

# Shell Variables

- Important to understand that export does not affect the parent process / shell, only children.
- **Example:**

**/cs2211/week4/setvar**

```
#!/bin/bash

MYVAR='This is the value of myvar in the
setvar script.'
export MYVAR
echo "MYVAR = $MYVAR"

SHELL=/bin/tcsh
echo "SHELL = $SHELL"
```

# Shell Variables

- Important to understand that **`export`** does not affect the parent process / shell, only children.
- **Output:**

```
[dservos5@cs2211b ~]$ MYVAR='Value of MYVAR in shell.'
[dservos5@cs2211b ~]$ echo $MYVAR
Value of MYVAR in shell.
[dservos5@cs2211b ~]$ echo $SHELL
/bin/sh
[dservos5@cs2211b ~]$ /cs2211/week4/setvar
MYVAR = This is the value of myvar in the setvar script.
SHELL = /bin/tcsh
[dservos5@cs2211b ~]$ echo $MYVAR
Value of MYVAR in shell.
[dservos5@cs2211b ~]$ echo $SHELL
/bin/sh
```

# Shell Variables

- Important to understand that **`export`** does not affect the parent process / shell, only children.
- **Output:**

```
[dservos5@cs2211b ~]$ MYVAR='Value of MYVAR in shell.'
[dservos5@cs2211b ~]$ echo $MYVAR
Value of MYVAR in shell.
[dservos5@cs2211b ~]$ echo $SHELL
/bin/sh
[dservos5@cs2211b ~]$ /cs2211/week4/setvar
MYVAR = This is the value of myvar in the setvar script.
SHELL = /bin/tcsh
[dservos5@cs2211b ~]$ echo $MYVAR
Value of MYVAR in shell.
[dservos5@cs2211b ~]$ echo $SHELL
/bin/sh
```

Value in shell before script is run

Value in script

Value in shell after script is run

# Shell Variables

- t
- .

**Value of $MYVAR and $SHELL did not change in the parent process/shell. export only affects child process/shell, even for environment variables.**

```
[dservos5@cs2211b ~]$ echo $MYVAR
Value of MYVAR in shell.
[dservos5@cs2211b ~]$ echo $SHELL
/bin/sh
[dservos5@cs2211b ~]$ /cs2211/week4/setvar
MYVAR = This is the value of myvar in the setvar script.
SHELL = /bin/tcsh
[dservos5@cs2211b ~]$ echo $MYVAR
Value of MYVAR in shell.
[dservos5@cs2211b ~]$ echo $SHELL
/bin/sh
```

# Shell Variables

Some useful shell variables:

| Variable | Example Value | Description |
|---|---|---|
| $PATH | /usr/local/bin:/usr/bin:. | Locations the shell searches for executables. |
| $HOME | /gaul/s1/student/1985/dservos5 | Your home directory. |
| $SHELL | /bin/bash | Your login shell. |
| $USER | dservos5 | Your username. |
| $PWD | /gaul/s1/student/1985/dservos5 | Your current working directory. |
| $SSH_TTY | /dev/pts/32 | Your current terminal device. |
| $HOSTNAME | cs2211b.gaul.csd.uwo.ca | The hostname of the server/computer. |
| $EDITOR | nano | Your default text editor. |
| $PS1 | [\u@\h \W]\$ | The formatting of your prompt. |
| $UID | 17789 | Your user id. |
| $LANG | en_US.UTF-8 | The system's language settings. |

# Shell Variables

- We can view a list of all variables using the **`set`** command with no arguments.
- **Example:**

```
[dservos5@cs2211b week4]$ set
BASH=/bin/bash
BASHOPTS=checkwinsize:cmdhist:expand_aliases:extquote:force_fignore
:histappend:hostcomplete:interactive_comments:login_shell:progcomp:
promptvars:sourcepath
BASH_ALIASES=()
BASH_ARGC=()
BASH_ARGV=()
BASH_CMDS=()
BASH_LINENO=()
BASH_SOURCE=()
BASH_VERSINFO=([0]="4" [1]="2" [2]="46" [3]="2" [4]="release"
[5]="x86_64-redhat-linux-gnu")
BASH_VERSION='4.2.46(2)-release'
COLUMNS=80
```

# Shell Variables

- We can view a list of just the environment variables using the **env** command with no arguments.
- **Example:**

```
[dservos5@cs2211b week4]$ env
XDG_SESSION_ID=2456
HOSTNAME=cs2211b.gaul.csd.uwo.ca
SELINUX_ROLE_REQUESTED=
TERM=xterm
SHELL=/bin/sh
HISTSIZE=1000
SSH_CLIENT=135.23.234.30 4132 22
SELINUX_USE_CURRENT_RANGE=
OLDPWD=/gaul/s1/student/1985/dservos5
SSH_TTY=/dev/pts/32
USER=dservos5
LS_COLORS=rs=0:di=01;34:ln=01;36:mh=00:pi=40;33:so=01;35:do=01;35:b
d=40;33;01:cd=40;33;01:or=40;31;01:mi=01;05;37;41:su=37;41:sg=30;43
:ca=30;41:tw=30;42:ow=34;42:st=37;44:ex=01;32:*.tar=01;31:*.tgz=01;
```

# Exit Status

- Every program, shell script, etc. executed by the shell or a shell script has an **exit status**.
- This **exit status** tells us if the program was successful and in the event of a failure, might give us a clue as to what went wrong.
- The exit status of a program is represented as an integer between 0 and 255.
- An **exit status of 0** indicates that the program exited **successfully**.
- An **exit status of 1 to 255** indicates that some kind of **failure** occurred. The value is normally some kind of error code to give us a clue as to what went wrong.

# Exit Status

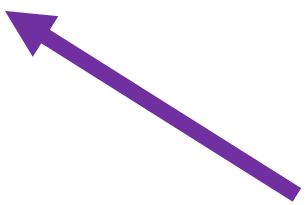
- We can access the exit status of the last run program using the special **$?** shell variable.
- **Examples:**

```
[dservos5@cs2211b ~]$ ls /cs2211
bin  week3  week4
[dservos5@cs2211b ~]$ echo $?
0
```

# Exit Status

- We can access the exit status of the last run program using the special **`$?`** shell variable.
- **Examples:**

```
[dservos5@cs2211b ~]$ ls /cs2211
bin  week3  week4
[dservos5@cs2211b ~]$ echo $?
0
```



**An exit code of 0 indicates that `ls` exited successfully.**

# Exit Status

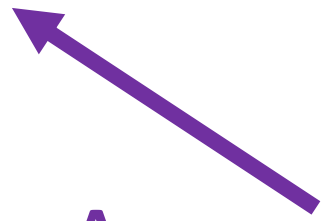
- We can access the exit status of the last run program using the special **$?** shell variable.
- **Examples:**

```
[dservos5@cs2211b ~]$ ls afilethatdoesnotexist
ls: cannot access afilethatdoesnotexist: No such file or directory
[dservos5@cs2211b ~]$ echo $?
2
```

# Exit Status

- We can access the exit status of the last run program using the special **$?** shell variable.
- **Examples:**

```
[dservos5@cs2211b ~]$ ls afilethatdoesnotexist
ls: cannot access afilethatdoesnotexist: No such file or directory
[dservos5@cs2211b ~]$ echo $?
2
```



**A non zero exit code indicates that `ls` failed.**

# Exit Status

- In many cases we can find the meaning of a non zero exit status in the man page for the command.
- For example, in the **`ls`** command's man page we find:

```
Exit status:
       0      if OK,
       1      if minor problems (e.g., cannot access
              subdirectory),
       2      if serious trouble (e.g., cannot access
              command-line argument).
```

# && and || Operators

- The logical operators **&&** and **||** allow us to perform some action based on the exit status returned by a command.
  - `cmd1 && cmd2` Execute `cmd2` only if `cmd1` was successful (zero exit status).
  - `cmd1 || cmd2` Execute cmd2 only if cmd1 fails (non zero exit status).

# && and || Operators

**Example:** Print “We found the file!” only if *file.txt* exits.

```
ls file.txt && echo 'We found the file!'
```

# && and || Operators

**Example:** Print “Pattern not found.” if *file.txt* does not contain the words cat or dog.

```
grep -E 'cat|dog' file.txt || echo 'Pattern not found.'
```

# && and || Operators

**Example:** Print “Pattern not found.” if *file.txt* does not contain the words cat or dog.

```
grep -E 'cat|dog' file.txt || echo 'Pattern not found.'
```

**grep returns the exit status 1 if it cannot find the pattern.**

# && and || Operators

**Example:** Print “Pattern not found.” if *file.txt* does not contain the words cat or dog.

```
grep -E 'cat|dog' file.txt || echo 'Pattern not found.'
```

**grep returns the exit status 1 if it can not find the pattern.**

**But it also returns the exit status 2 if it can not find the file. This means that the above command would also print “Pattern not found” if file.txt does not exist.**

# Shell Scripts: Part 1

# Shell Scripts

- A shell script is a *text file* containing commands to be run sequentially by the shell.
- Shell scripts contain either *UNIX commands* as we would issue them from the command line or *flow control statements* (IF, WHILE, FOR, etc.).
- Commands are executed by the shell in the order they appear or in an order determined by flow control statements.
- Different shells (e.g. bash, csh, tcsh, etc.) have slightly different syntax and flow control structures.
- For this course, we will be writing **bash** shell scripts.

# Shell Scripts

- As shell scripts are text files, you can create them with any text editor (nano, vim, emacs, etc).
- They **do not** require a file extension, but it is common convention to give them a *.sh* or *.bash* extension to denote them as shell scripts.
- Shell scripts are **interpreted** by the shell and **not compiled** (unlike C code).
- We use shell scripts to **avoid repetition**, **automate tasks** (e.g. backup files regularly) or **create simple programs** that mainly use other UNIX commands.

# Shell Scripts

## #!

- The first line of a shell script specifies what shell it should be interpreted by (or “run” in).
- The line has the format:

```
#!/path/to/shell/executable
```

- For example, if we want our shell script to be a bash script, the first line should be:

```
#!/bin/bash
```

# Shell Scripts

#!

- The first line of a shell script specifies what shell it should be interpreted by (or “run” in).
- The line has the format:

**On the course server */bin* is a symbolic link to the */usr/bin* directory.**

- For example, if we want our shell script to be a bash script, the first line should be:

#!/bin/bash

# Shell Scripts

#!

- If this line is omitted (or not the first line of the file), the current shell will be used instead of the one specified.
- This can be a problem as the syntax of shell scripts can vary between shells.
- In assignments, quizzes and exams it is expected that you include this line in all of your shell scripts.

# Shell Scripts

## Running Shell Scripts

### Method One

- If the script is in our PATH we can run by simply giving its file name:

```
[dservos5@cs2211b week4]$ helloworld.sh
Hello World!
```

- If the script is not in our PATH we need to provide an absolute or relative path to the script (even if it is in our current working directory):

```
[dservos5@cs2211b week4]$ ./helloworld.sh
Hello World!
```

# Shell Scripts

## Running Shell Scripts

### Method One

- For this method to work, the script must have the executable permission set (+x). Recall that you can set this permission on a file for yourself with:

```
chmod +x file
```

- When we run a script with this method, it is interpreted by the shell specified by the **#!** line or the current shell if the **#!** line is omitted.

# Shell Scripts

## Running Shell Scripts

### Method Two

- We can also run a shell script by calling the shell with the script's file name as an argument:

```
[dservos5@cs2211b week4]$ bash helloworld.sh
Hello World!
```

- This method ignores the **#!** line and runs the script in the called shell.
- When executing a shell script like this, the execute permission is not required.

# Shell Scripts

## Running Shell Scripts

**Which method is best practice? Why?**

# Shell Scripts

## Simple Example

**/cs2211/week4/simple.sh**

```
#!/bin/bash
# A simple shell script that prints the date,
# current user and login shell.

echo "Hello $USER."
echo "Today is: `date`."
echo "Your shell today is $SHELL."
```

# Shell Scripts

## Simple Example

**/cs2211/week4/simple.sh**

```
#!/bin/bash
# A simple shell script that prints the date,
# current user and login shell
echo "Hello $USER."
echo "Today is: `date`."
echo "Your shell today is $SHELL."
```

**#! Line that specifies the shell to use.**

# Shell Scripts

## Simple Example

**/cs2211/week4/simple.sh**

```
#!/bin/bash
# A simple shell script that prints the date,
# current user and login shell.
```

**Comment that is ignored by the shell.**

**Comments start with a #**

**Use comments to document your shell scripts.**

# Shell Scripts

## Simple Example

**/cs2211/week4/simple.sh**

```
#!/bin/bash
# A simple shell script that prints the date,
# current user and login shell.

echo "Hello $USER."
echo "Today is: `date`."
echo "Your shell today is $SHELL."
```

**These commands work just like if we called them in the shell.**

# Shell Scripts

## Simple Example: Output

```
[dservos5@cs2211b ~]$ /cs2211/week4/simple.sh
Hello dservos5.
Today is: Thu Feb  1 12:10:00 EST 2018.
Your shell today is /bin/bash.
```

# Shell Scripts

## Reading From Standard Input

We can use the **`read`** command to read values from standard input into a shell variable.

```
read [options] varname …
```

# Shell Scripts

## Reading From Standard Input

**Simple `read` Example:**

**/cs2211/week4/helloname.sh**

```
#!/bin/bash

echo -n "Input your name: "
read name

echo "Hello $name. How are you?"
```

# Shell Scripts

## Reading From Standard Input

S

-n option to echo causes no new line to be printed. Input will be taken on same line as output.

/cs2211/week4/hellohame.sh

```
#!/bin/bash

echo -n "Input your name: "
read name

echo "Hello $name. How are you?"
```

# Shell Scripts

## Reading From Standard Input

**S**

**We are reading one value into the shell variable $name from standard input.**

**/cs2211/week4/hellohame.sh**

```
#!/bin/bash

echo -n "Input your name: "
read name

echo "Hello $name. How are you?"
```

# Shell Scripts

## Reading From Standard Input

**Simple `read` Example Output:**

```
[dservos5@cs2211b ~]$ /cs2211/week4/helloname.sh
Input your name: Daniel
Hello Daniel. How are you?
```

# Shell Scripts

## Reading From Standard Input

**Simple `read` Example Output:**

```
[dservos5@cs2211b ~]$ /cs2211/week4/helloname.sh
Input your name: Daniel
Hello Daniel. How are you?
```

**Type in from keyboard (standard input).**

**Input is ended by line break (pressing enter).**

# Shell Scripts

## Reading From Standard Input

**Example reading multiple values:**

**/cs2211/week4/multiread.sh**

```
#!/bin/bash

echo -n "Input your name, age and location: "
read name age loc

echo "Hello $name."
echo "You are $age years old."
echo "You are from $loc."
```

# Shell Scripts

## Reading From Standard Input

**Example reading multiple values:**

**/cs2211/week4/multiread.sh**

```
#!/bin/bash

echo -n "Input your name, age and location: "
read name age loc

echo "Hello $name."
echo "You are $age years old."
echo "You are from $loc."
```

**Each argument to read is a new value that will be read into a shell variable.**

# Shell Scripts

## Reading From Standard Input

**multiread.sh Output: Case 1**

```
[dservos5@cs2211b ~]$ /cs2211/week4/multiread.sh
Input your name, age and location: Daniel 32 London
Hello Daniel.
You are 32 years old.
You are from London.
```

# Shell Scripts

## Reading From Standard Input

**multiread.sh Output: Case 2**

```
[dservos5@cs2211b ~]$ /cs2211/week4/multiread.sh
Input your name, age and location: Daniel Servos 32 London
Hello Daniel.
You are Servos years old.
You are from 32 London.
```

**Entered too many values!**

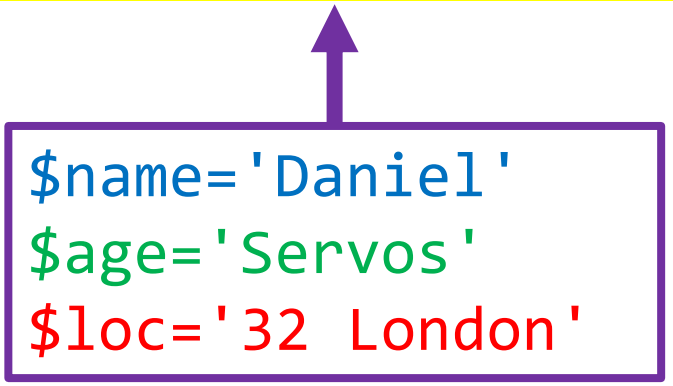
**Spaces or tabs separate values when using read.**

# Shell Scripts

## Reading From Standard Input

### multiread.sh Output: Case 2

```
[dservos5@cs2211b ~]$ /cs2211/week4/multiread.sh
Input your name, age and location: Daniel Servos 32 London
Hello Daniel.
You are Servos years old.
You are from 32 London.
```




```
read name age loc
```

# Shell Scripts

## Reading From Standard Input

### multiread.sh Output: Case 2

```
[dservos5@cs2211b ~]$ /cs2211/week4/multiread.sh
Input your name, age and location: Daniel Servos 32 London
Hello Daniel.
You are Servos years old.
You are from 32 London.
```

```
$name='Daniel'
$age='Servos'
$loc='32 London'
```

```
read name age loc
```

**Last argument to read gets all extra input.**

# Shell Scripts

## Reading From Standard Input

### multiread.sh Output: Case 3

```
[dservos5@cs2211b ~]$ /cs2211/week4/multiread.sh
Input your name, age and location: Daniel\ Servos 32 London
Hello Daniel Servos.
You are 32 years old.
You are from London.
```

# Shell Scripts

## Reading From Standard Input

### multiread.sh Output: Case 3

```
[dservos5@cs2211b ~]$ /cs2211/week4/multiread.sh
Input your name, age and location: Daniel\ Servos 32 London
Hello Daniel Servos.
You are 32 years old.
You are from London.
```

**Space is escaped and “Daniel Servos” is read in as one string.**

# Shell Scripts

## Reading From Standard Input

**More Complex Example:**

Write a shell script to count how many times a user is logged into the system. The script should take a username from standard input.

# Shell Scripts

## Reading From Standard Input

**More Complex Example:**

**/cs2211/week4/usercount.sh**

```
#!/bin/bash

read -p "Input a username: " username
count=`who | grep -c "^$username\s"`

echo "$username is logged in $count times."
```

# Shell Scripts

## Reading From Standard Input

**-p option to read prints a prompt for us, so we do not have to use echo.**

**/cs2211/week4/usercount.sh**

```
#!/bin/bash

read -p "Input a username: " username
count=`who | grep -c "^$username\s"`

echo "$username is logged in $count times."
```

# Shell Scripts

## Reading From Standard Input

**More Complex Example:**

**/cs2211/week4/usercount.sh**

```
#!/bin/bash

read -p "Input a username: " username
count=`who | grep -c "^$username\s"`

echo "$username is logged in $count times."
```

**Result of the command `who | grep -c "^$username\s"` is stored in the variable `$count`.**

# Shell Scripts

## Reading From Standard Input

**usercount.sh Output:**

```
[dservos5@cs2211b ~]$ /cs2211/week4/usercount.sh
Input a username: dservos5
dservos5 is logged in 1 times.
```

# Shell Scripts

## Arithmetic Using `expr`

- Most shells do not have computing or string handling features built in (bash is an exception that we will see later).
- To do things like basic arithmetic you need to use a program like **expr** or **bc**.
- **expr** only handles integer arithmetic (decimals are truncated).
- **bc** supports floating point (decimal) arithmetic, but is not always installed on the system.

# Shell Scripts

## Arithmetic Using `expr`

- The **`expr`** command takes a mathematical or logical expression as its arguments and returns the result via standard output.
- Valid arithmetic operators are: +, -, *, /, %
- Valid comparison operators are: >,<,>=,<=,==,!=
- Valid logic operators are: &, |
- Parentheses, **(** and **)**,can be used to control order of operations.

# Shell Scripts

## Arithmetic Using expr

**Example Use on Command Line:**

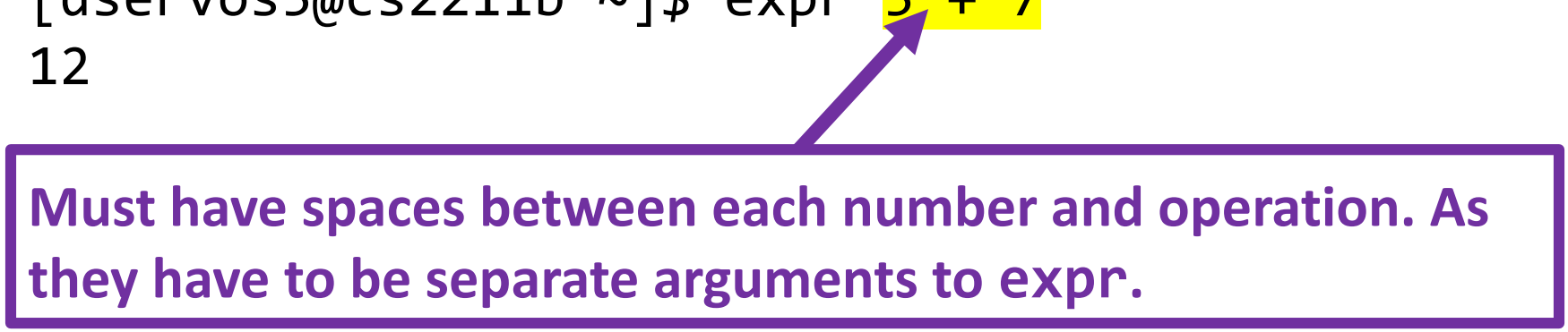
```
[dservos5@cs2211b ~]$ expr 5 + 7
12

[dservos5@cs2211b ~]$ expr 100 - 10 + 4
94

[dservos5@cs2211b ~]$ expr \( 50 - 10 \) \* 2
80

[dservos5@cs2211b ~]$ expr 5 / 2
2
```

# Shell Scripts

## Arithmetic Using expr

**Example Use on Command Line:**

```
[dservos5@cs2211b ~]$ expr 5 + 7
12
```




```
[dservos5@cs2211b ~]$ expr \( 50 - 10 \) \* 2
80

[dservos5@cs2211b ~]$ expr 5 / 2
2
```

# Shell Scripts

## Arithmetic Using expr

**Example Use on Command Line:**

**We have to escape any special characters or shell wild cards.**

**We cannot use quoting. Why?**

```
94

[dservos5@cs2211b ~]$ expr \( 50 - 10 \) \* 2
80

[dservos5@cs2211b ~]$ expr 5 / 2
2
```

# Shell Scripts

## Arithmetic Using expr

**Example Use on Command Line:**

```
[dservos5@cs2211b ~]$ expr 5 + 7
12

[dservos5@cs2211b ~]$ expr 100 - 10 + 4
94
```

**expr does integer arithmetic. Result of division is always an integer. Decimal points are truncated.**

```
80

[dservos5@cs2211b ~]$ expr 5 / 2
2
```

# Shell Scripts

## Arithmetic Using `expr`

**Example in Shell Script:**

Write a shell script that reads in three numbers from standard input and finds the average of these numbers.

# Shell Scripts

## Arithmetic Using `expr`

**Example in Shell Script:**

**/cs2211/week4/avg.sh**

# Shell Scripts

## Arithmetic Using `expr`

**#! Line specifying that script should be run in bash shell.**

**/cs2211/week4/avg.sh**

```
#!/bin/bash
```

# Shell Scripts

## Arithmetic Using expr

**Output a prompt saying "Input three numbers:" and read in three values into $n1, $n2 and $n3.**

**/cs2211/week4/avg.sh**

```
#!/bin/bash

read -p "Input three numbers: " n1 n2 n3
```

# Shell Scripts

## Arithmetic Using expr

**Use expr to take the sum of $n1, $n2 and $n3 and store the result in $sum.**

**/cs2211/week4/avg.sh**

```
#!/bin/bash

read -p "Input three numbers: " n1 n2 n3

sum=`expr $n1 + $n2 + $n3`
```

**Remember that back quotes make the shell execute the command in the quotes and return the result. Without the back quotes we would get an error**

# Shell Scripts

## Arithmetic Using expr

**Use expr to divide the value of $sum by 3. Store the result in $avg.**

**/cs2211/week4/avg.sh**

```
#!/bin/bash

read -p "Input three numbers: " n1 n2 n3

sum=`expr $n1 + $n2 + $n3
avg=`expr $sum / 3`
```

# Shell Scripts

## Arithmetic Using expr

**Output the values of $n1, $n2, and $n3 as well as the average we calculated stored in $avg.**

**/cs2211/week4/avg.sh**

```
#!/bin/bash

read -p "Input three numbers: " n1 n2 n3

sum=`expr $n1 + $n2 + $n3`
avg=`expr $sum / 3`

echo "Average of $n1, $n2, and $n3 is: $avg"
```

# Shell Scripts

## Arithmetic Using `expr`

**Example in Shell Script: Output**

```
[dservos5@cs2211b ~]$ /cs2211/week4/avg.sh
Input three numbers: 42 -5 101
Average of 42, -5, and 101 is: 46
```