Casey Bladow
CSIS 360 - Spring 2015
Assignment 10 - 25 points
Due Tuesday, April 21

Follow the **Creating Assignments** handout while performing the below steps. The purpose of this assignment is to ...

If any of you find that I am misusing or have incorrect terminology in any of my descriptions, please drop me a note with an explanation. I'm learning too!

1. Install the *tcsh* shell on your Ubuntu box using your favored method of installation.

2. Read chapter 9 (The Bourn Again Shell) while at the keyboard experimenting with the commands as you read the chapter. My goodness it's a long chapter! The below activities will let you know what you are responsible for on the final test. Since this is a long chapter, it may be tempting to skip the reading and just do the activities. Instead, it would be better to limit the experimenting as you read through it. To reduce the reading, feel free to quickly browse or skip the following
    - The Readline Library - pages 340-346
    - The Functions section through the end of the chapter - pages 349-365. I plan on having you do functions in a later assignment. One of the topics, Brace Expansion on page 358 you've already used... file{1..50}, you could also use specific ones as in file{A,B,C}.

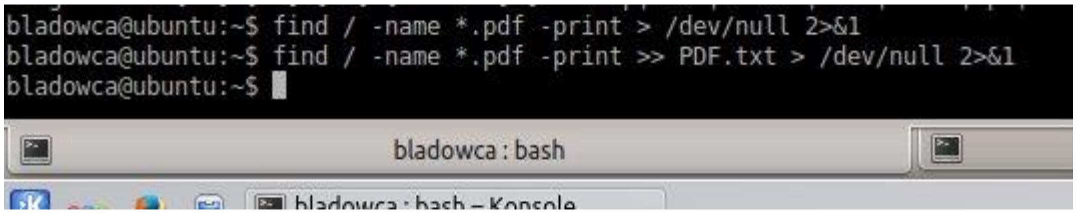
3. Chapter 9 activities
    - In assignment 9 (chapter 7), you learned about redirecting standard input and standard output. This short activity gives a quick demonstration of redirecting standard error.
        - Open a terminal window and display every PDF document on your Ubuntu box.

          ```
          find / -name *.pdf -print
          ```

        - Note the large number of Permission denied errors.
        - 0 is standard input, 1 is standard output, and 2 is standard error. Repeat the same command but this time redirect standard error

(*2>*) to the data sink you learned about in chapter 7 that makes data disappear (*/dev/null*).

- Repeat the same command but this time redirect standard output to the file PDF.txt and standard error to */dev/null*.
- Make sure results for the last two commands are displayed within the terminal window (if you made command mistakes that caused the results to scroll off the screen, redo them) then ***take a window grab of the terminal window and include it in your assignment 10 document***



```
bladowca@ubuntu:~$ find / -name *.pdf -print > /dev/null 2>&1
bladowca@ubuntu:~$ find / -name *.pdf -print >> PDF.txt > /dev/null 2>&1
bladowca@ubuntu:~$ 
```

- This activity will make sure you have an understanding of the different shells when executing shell scripts as the associated syntax can differ. This is not an attempt to get you to use different shells for different things, only to make you aware of the different shells that are available and be able to debug shell scripts to work on the shell you are using. As far as I'm concerned, I'd recommend sticking with *bash* shell which is the default.
  - Open a terminal window and resize it vertically so that it has maximum height from the top to bottom of the screen (more room for me to see your results when you do the window grab).
  - Install tcsh if you haven't already.
  - Create the file *shellscript* using the *vim* editor containing the following (this is similar, not exact, to the example on page 302 of your textbook):

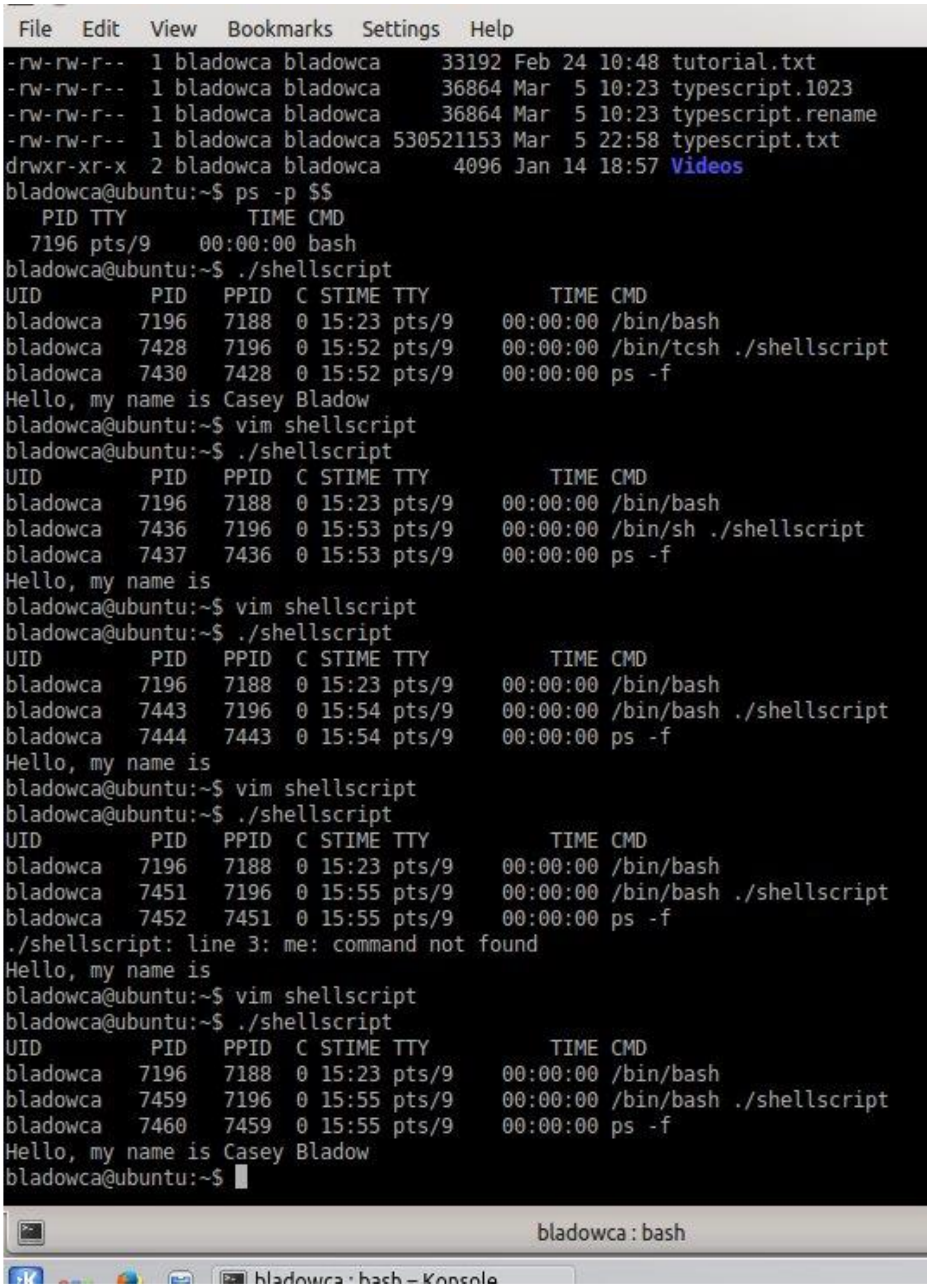
```
#!/bin/tcsh
ps -f
set me = "Dan Brekke" (substitute your name)
echo "Hello, my name is $me"
```

  - Execute *shellscript* with the command *source ./shellscript* (alternatively you could replace the word *source* with a . (dot) as you learned on page 296).
  - Now attempt to execute it directly with the command *./shellscript* and note that permission was denied.

- Type the command *ls -l* and note that the file does not have the execution bit set in the file protections.
- Change the protection on the file so that it is an executable (you learned about the *chmod* utility in in chapter 6).
- Type the command *ps -p $$* and note that your shell (unless you've changed it) is bash.
- Execute the script with the command *./shellscript* noting the shell that executed the script is *tcsh*
- Edit the file again using the *vim* editor changing the shell from *tcsh* to *sh* then execute the script. Note the shell that executed the script is *sh* and that it didn't execute correctly (doesn't display your name). That's because the *tcsh* shell has different syntax than the *sh* shell.
- Edit the file again using the *vim* editor changing the shell from *sh* to *bash* then execute the script. Note the results.
- Edit the file again and change the third line to:

-
  ```
  me="Dan Brekke"
  ```

substituting your name. Note there is not any blanks in the line (except between your first and last name). Run the shell script again. ***Take a window grab of the terminal window and include it in your assignment 10 document***

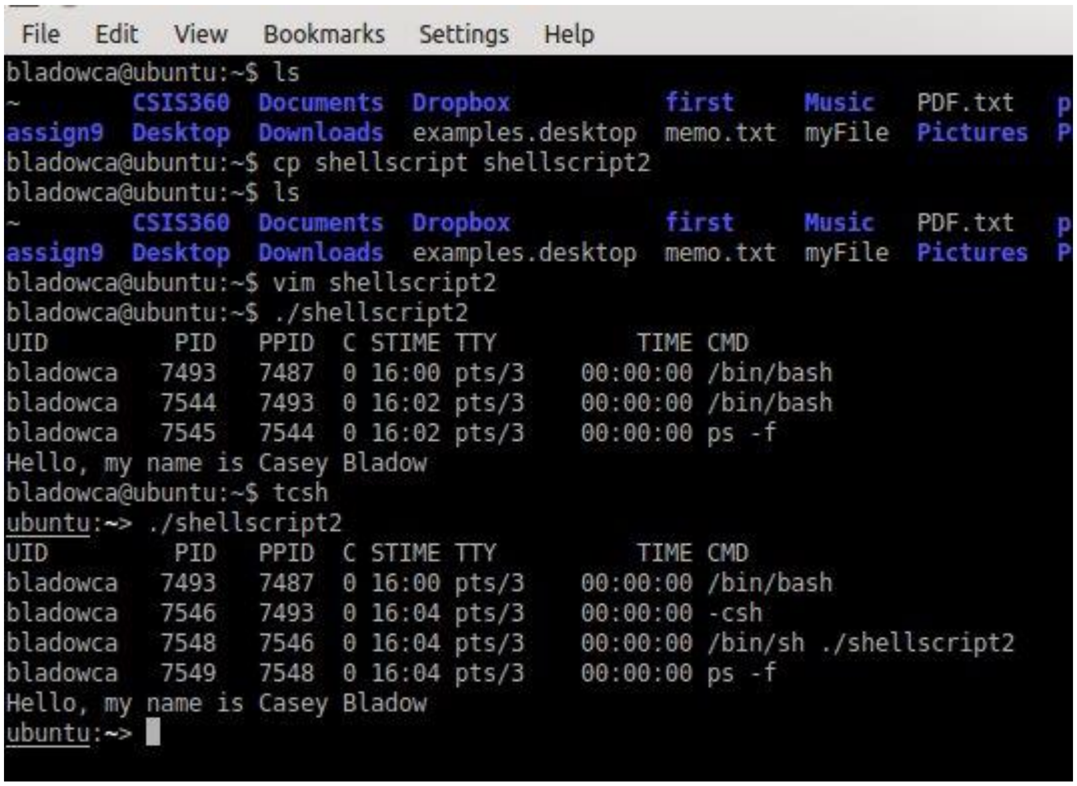

File Edit View Bookmarks Settings Help
bladowca : bash
bladowca : bash – Konsole

```
-rw-rw-r--  1 bladowca bladowca     33192 Feb 24 10:48 tutorial.txt
-rw-rw-r--  1 bladowca bladowca     36864 Mar  5 10:23 typescript.1023
-rw-rw-r--  1 bladowca bladowca     36864 Mar  5 10:23 typescript.rename
-rw-rw-r--  1 bladowca bladowca 530521153 Mar  5 22:58 typescript.txt
drwxr-xr-x  2 bladowca bladowca      4096 Jan 14 18:57 Videos
bladowca@ubuntu:~$ ps -p $$
  PID TTY          TIME CMD
 7196 pts/9    00:00:00 bash
bladowca@ubuntu:~$ ./shellscript
UID        PID  PPID  C STIME TTY          TIME CMD
bladowca  7196  7188  0 15:23 pts/9    00:00:00 /bin/bash
bladowca  7428  7196  0 15:52 pts/9    00:00:00 /bin/tcsh ./shellscript
bladowca  7430  7428  0 15:52 pts/9    00:00:00 ps -f
Hello, my name is Casey Bladow
bladowca@ubuntu:~$ vim shellscript
bladowca@ubuntu:~$ ./shellscript
UID        PID  PPID  C STIME TTY          TIME CMD
bladowca  7196  7188  0 15:23 pts/9    00:00:00 /bin/bash
bladowca  7436  7196  0 15:53 pts/9    00:00:00 /bin/sh ./shellscript
bladowca  7437  7436  0 15:53 pts/9    00:00:00 ps -f
Hello, my name is
bladowca@ubuntu:~$ vim shellscript
bladowca@ubuntu:~$ ./shellscript
UID        PID  PPID  C STIME TTY          TIME CMD
bladowca  7196  7188  0 15:23 pts/9    00:00:00 /bin/bash
bladowca  7443  7196  0 15:54 pts/9    00:00:00 /bin/bash ./shellscript
bladowca  7444  7443  0 15:54 pts/9    00:00:00 ps -f
Hello, my name is
bladowca@ubuntu:~$ vim shellscript
bladowca@ubuntu:~$ ./shellscript
UID        PID  PPID  C STIME TTY          TIME CMD
bladowca  7196  7188  0 15:23 pts/9    00:00:00 /bin/bash
bladowca  7451  7196  0 15:55 pts/9    00:00:00 /bin/bash ./shellscript
bladowca  7452  7451  0 15:55 pts/9    00:00:00 ps -f
./shellscript: line 3: me: command not found
Hello, my name is
bladowca@ubuntu:~$ vim shellscript
bladowca@ubuntu:~$ ./shellscript
UID        PID  PPID  C STIME TTY          TIME CMD
bladowca  7196  7188  0 15:23 pts/9    00:00:00 /bin/bash
bladowca  7459  7196  0 15:55 pts/9    00:00:00 /bin/bash ./shellscript
bladowca  7460  7459  0 15:55 pts/9    00:00:00 ps -f
Hello, my name is Casey Bladow
bladowca@ubuntu:~$
```

- Type the command *cat /etc/shells* which will give you the available shells.
- The *chsh* utility (page 293) will permanently change your default shell. The default shell *sh* will be used if the *#!/bin/shellname* line is absent from a shell script. Changing a shell with *chsh*takes

effect only at the next login. An immediate change can be accomplished by typing the shell name. For example, *tcsh*, or *sh*, or *bash*, or any of the others you saw in */etc/shells*. Try a few of these. Type *ps -p $$* to see the current shell that you are using. Type *ps* to see the shells that you have started. When you press CONTROL-D or type *exit* you will return to the previous shell.

- Exit out of all the shells then open a new terminal window.
- Make a copy of the file *shellscript* to *shellscript2*.
- Using *vim*, edit the file *shellscript2* and remove the first line *#!/etc/bash*
- Run *shellscript2* with the command *./shellscript2* and note the results.
- Next type *tcsh* to change your shell, then run *shellscript2* again and note that it was run by the default shell *sh* even though you were using the *tcsh* shell. Moral of the story? It's good to use the first line *#!/bin/shellname* in shell scripts. ***Take a window grab of the terminal window and include it in your assignment 10 document***



```
bladowca@ubuntu:~$ ls
~        CSIS360  Documents  Dropbox           first     Music   PDF.txt   p
assign9  Desktop  Downloads  examples.desktop  memo.txt  myFile  Pictures  P
bladowca@ubuntu:~$ cp shellscript shellscript2
bladowca@ubuntu:~$ ls
~        CSIS360  Documents  Dropbox           first     Music   PDF.txt   p
assign9  Desktop  Downloads  examples.desktop  memo.txt  myFile  Pictures  P
bladowca@ubuntu:~$ vim shellscript2
bladowca@ubuntu:~$ ./shellscript2
UID        PID  PPID  C STIME TTY          TIME CMD
bladowca  7493  7487  0 16:00 pts/3    00:00:00 /bin/bash
bladowca  7544  7493  0 16:02 pts/3    00:00:00 /bin/bash
bladowca  7545  7544  0 16:02 pts/3    00:00:00 ps -f
Hello, my name is Casey Bladow
bladowca@ubuntu:~$ tcsh
ubuntu:~> ./shellscript2
UID        PID  PPID  C STIME TTY          TIME CMD
bladowca  7493  7487  0 16:00 pts/3    00:00:00 /bin/bash
bladowca  7546  7493  0 16:04 pts/3    00:00:00 -csh
bladowca  7548  7546  0 16:04 pts/3    00:00:00 /bin/sh ./shellscript2
bladowca  7549  7548  0 16:04 pts/3    00:00:00 ps -f
Hello, my name is Casey Bladow
ubuntu:~>
```

- This activity will make sure you have an understanding of separating and grouping commands.

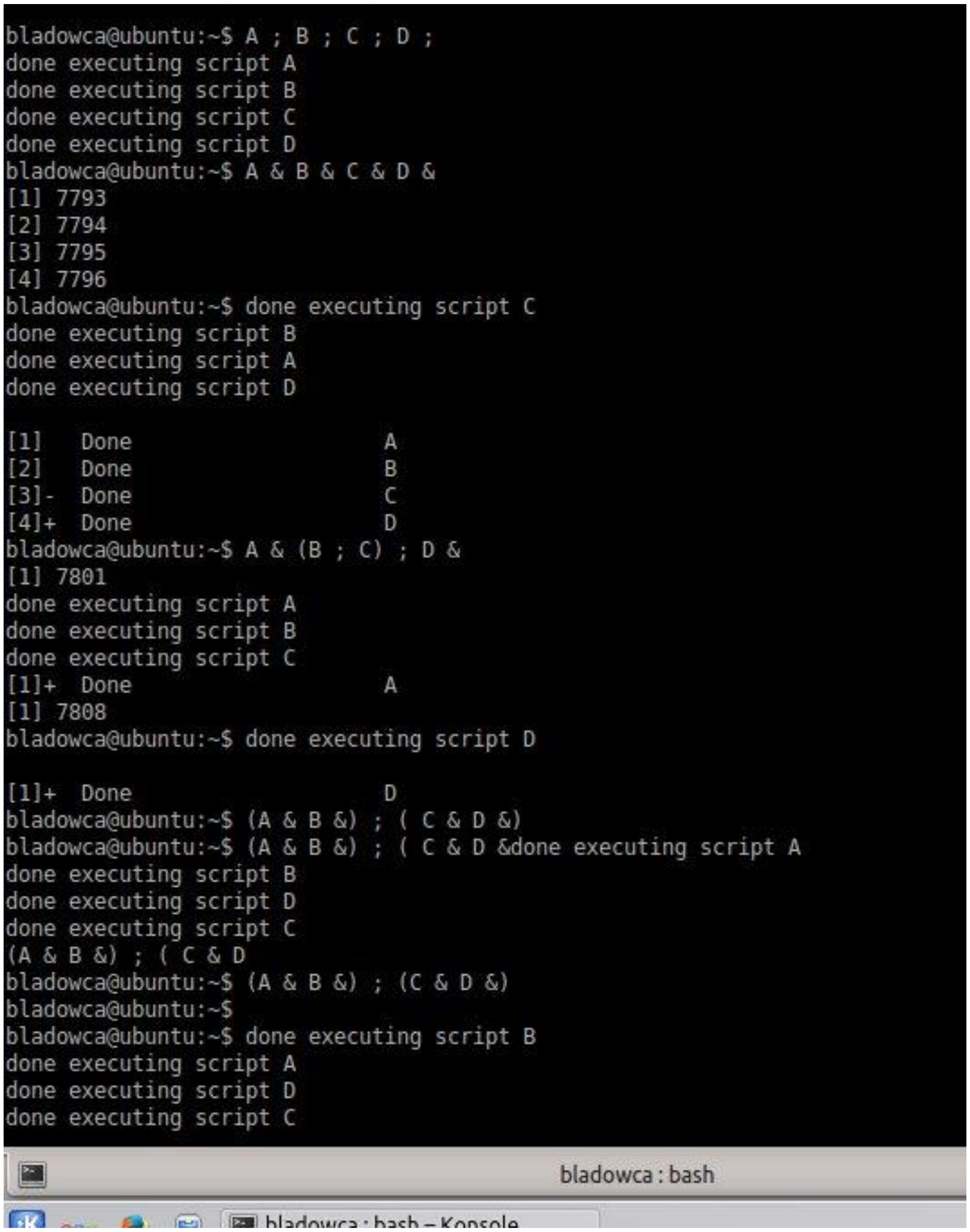
- Open a terminal window and resize it vertically so that it has maximum height from the top to bottom of the screen (more room for me to see your results when you do the window grab).
- Using the *vim* editor, create 4 files, *A*, *B*, *C*, and *D*. The contents of file *A* is as follows. Change the last line accordingly for each of the other files.
-

```
#!/bin/bash
sleep 10
echo "done executing script A"
```

- Make each of the files executable.
- If your working directory is not part of the PATH so that you can run executable images or script files from your directory, give the command that will temporarily add your working directory to the PATH (you gave the answer to this question in assignment 9). Once your working directory is part of the PATH, you can give the command *A* instead of *./A*
- Before you execute each of the below commands, think about what will happen and predict the results. After you see the results, make sure you understand why it was so.
-

```
A ; B ; C ; D ;
A & B & C & D &
A & (B ; C) ; D &
(A & B &) ; (C & D &)
(A ; B) & (C ; D)
A | B | C | D  (this doesn't make much sense
but what do you think it will do?)
```

- ***Do a window grab of the terminal window and include it in your assignment 10 document.***



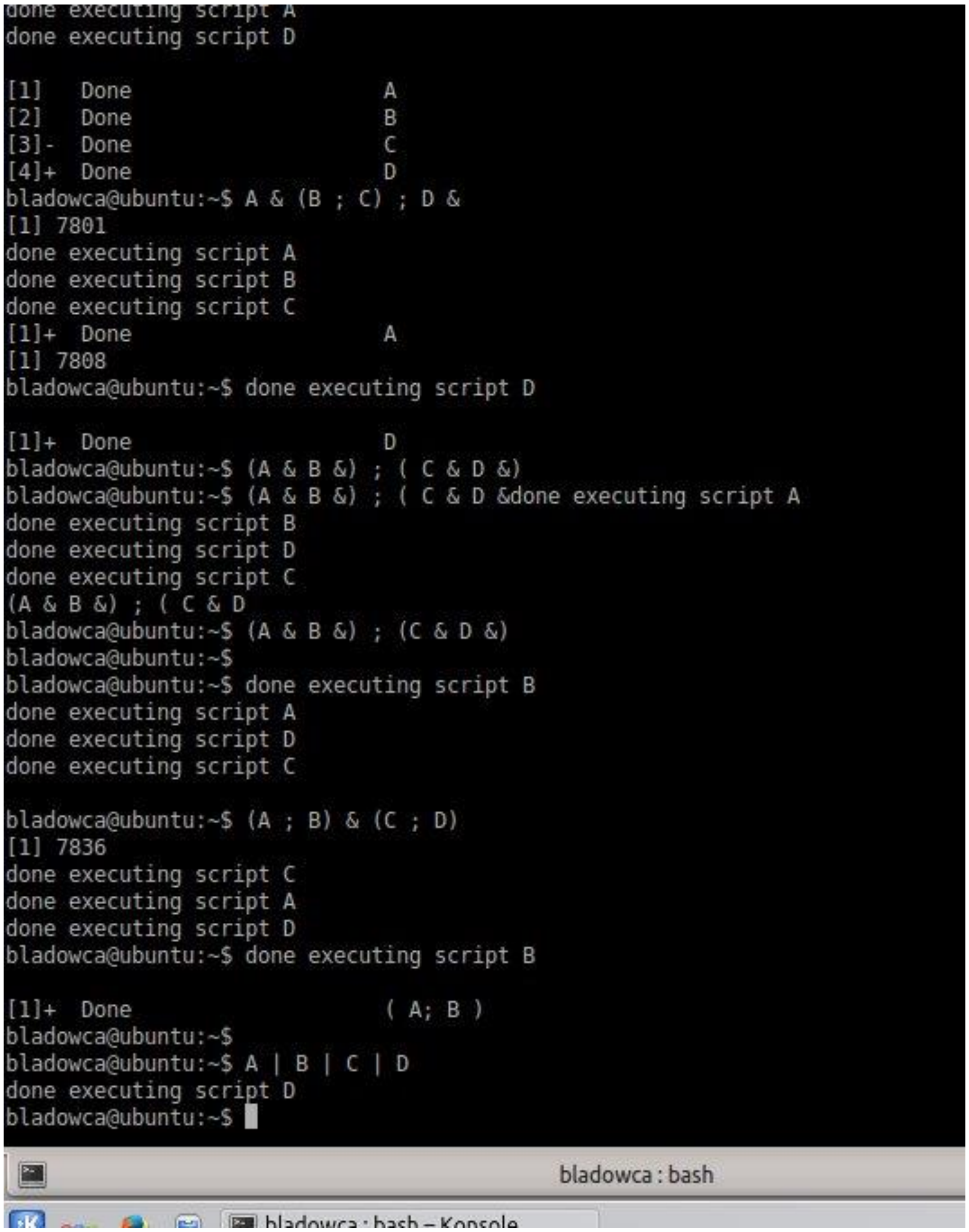
```
bladowca@ubuntu:~$ A ; B ; C ; D ;
done executing script A
done executing script B
done executing script C
done executing script D
bladowca@ubuntu:~$ A & B & C & D &
[1] 7793
[2] 7794
[3] 7795
[4] 7796
bladowca@ubuntu:~$ done executing script C
done executing script B
done executing script A
done executing script D

[1]   Done                    A
[2]   Done                    B
[3]-  Done                    C
[4]+  Done                    D
bladowca@ubuntu:~$ A & (B ; C) ; D &
[1] 7801
done executing script A
done executing script B
done executing script C
[1]+  Done                    A
[1] 7808
bladowca@ubuntu:~$ done executing script D

[1]+  Done                    D
bladowca@ubuntu:~$ (A & B &) ; ( C & D &)
bladowca@ubuntu:~$ (A & B &) ; ( C & D &done executing script A
done executing script B
done executing script D
done executing script C
(A & B &) ; ( C & D
bladowca@ubuntu:~$ (A & B &) ; (C & D &)
bladowca@ubuntu:~$
bladowca@ubuntu:~$ done executing script B
done executing script A
done executing script D
done executing script C
```

bladowca : bash

bladowca : bash – Konsole



```
done executing script A
done executing script D

[1]   Done                    A
[2]   Done                    B
[3]-  Done                    C
[4]+  Done                    D
bladowca@ubuntu:~$ A & (B ; C) ; D &
[1] 7801
done executing script A
done executing script B
done executing script C
[1]+  Done                    A
[1] 7808
bladowca@ubuntu:~$ done executing script D

[1]+  Done                    D
bladowca@ubuntu:~$ (A & B &) ; ( C & D &)
bladowca@ubuntu:~$ (A & B &) ; ( C & D &done executing script A
done executing script B
done executing script D
done executing script C
(A & B &) ; ( C & D
bladowca@ubuntu:~$ (A & B &) ; (C & D &)
bladowca@ubuntu:~$
bladowca@ubuntu:~$ done executing script B
done executing script A
done executing script D
done executing script C

bladowca@ubuntu:~$ (A ; B) & (C ; D)
[1] 7836
done executing script C
done executing script A
done executing script D
bladowca@ubuntu:~$ done executing script B

[1]+  Done                    ( A; B )
bladowca@ubuntu:~$
bladowca@ubuntu:~$ A | B | C | D
done executing script D
bladowca@ubuntu:~$ 
```

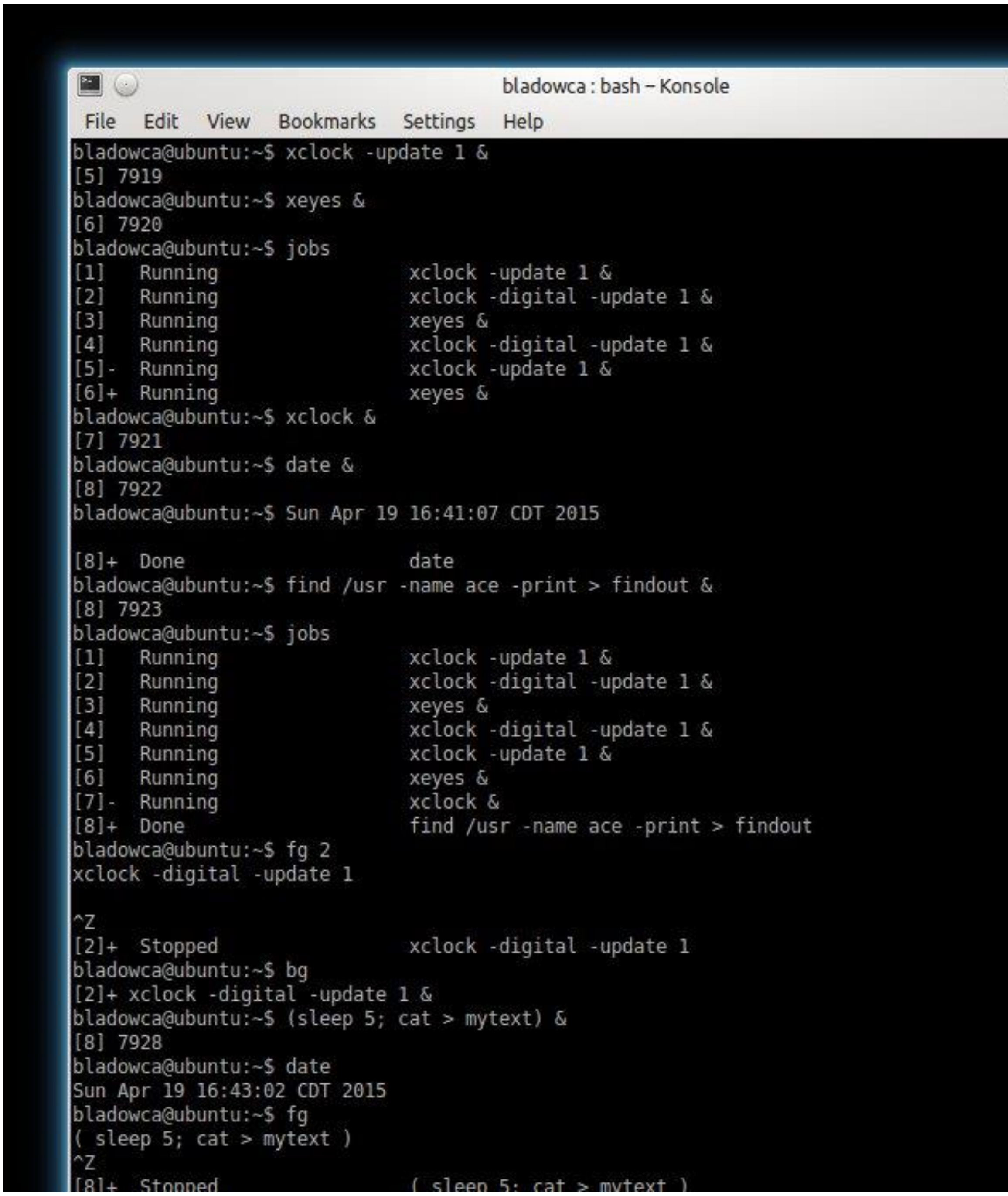
- Change the sleep times in the files to different values and experiment on your own.

- This activity will make sure you have an understanding of foreground and background jobs

- Open a terminal window and resize it vertically so that it has maximum height from the top to bottom of the screen (more room for me to see your results when you do the window grab).
- Type the following commands:

```
xclock -update 1 &
xclock -digital -update 1 &
xeyes &
jobs
```

- Move each of these windows away from each other so that they don't overlap and are not covering your terminal window.
- Experiment with the commands on pages 308-309 (*bg*, *fg*, and CONTROL-Z) until you get a good understanding of how they work. Note that after you typed the command *jobs* above, there was a + sign after job [3]. This means that if you typed the command *fg* without a job number, it would bring job 3 to the foreground. Notice the effect when you suspend any of these jobs (the clock stops or the eyes stop moving).
- ***Do a window grab of the terminal window that shows all your experimenting and include it in your assignment 10 document.***


bladowca : bash – Konsole
File Edit View Bookmarks Settings Help

```
bladowca@ubuntu:~$ xclock -update 1 &
[5] 7919
bladowca@ubuntu:~$ xeyes &
[6] 7920
bladowca@ubuntu:~$ jobs
[1]   Running                 xclock -update 1 &
[2]   Running                 xclock -digital -update 1 &
[3]   Running                 xeyes &
[4]   Running                 xclock -digital -update 1 &
[5]-  Running                 xclock -update 1 &
[6]+  Running                 xeyes &
bladowca@ubuntu:~$ xclock &
[7] 7921
bladowca@ubuntu:~$ date &
[8] 7922
bladowca@ubuntu:~$ Sun Apr 19 16:41:07 CDT 2015

[8]+  Done                    date
bladowca@ubuntu:~$ find /usr -name ace -print > findout &
[8] 7923
bladowca@ubuntu:~$ jobs
[1]   Running                 xclock -update 1 &
[2]   Running                 xclock -digital -update 1 &
[3]   Running                 xeyes &
[4]   Running                 xclock -digital -update 1 &
[5]   Running                 xclock -update 1 &
[6]   Running                 xeyes &
[7]-  Running                 xclock &
[8]+  Done                    find /usr -name ace -print > findout
bladowca@ubuntu:~$ fg 2
xclock -digital -update 1

^Z
[2]+  Stopped                 xclock -digital -update 1
bladowca@ubuntu:~$ bg
[2]+ xclock -digital -update 1 &
bladowca@ubuntu:~$ (sleep 5; cat > mytext) &
[8] 7928
bladowca@ubuntu:~$ date
Sun Apr 19 16:43:02 CDT 2015
bladowca@ubuntu:~$ fg
( sleep 5; cat > mytext )
^Z
[8]+  Stopped                 ( sleep 5; cat > mytext )
```

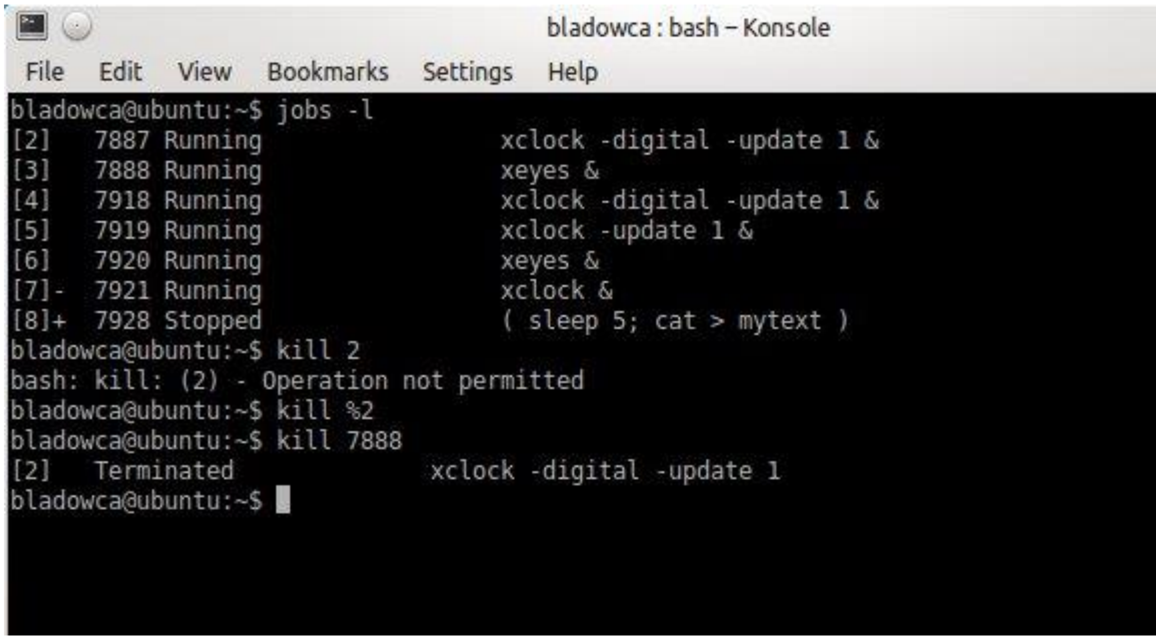
- Bring job 1 to the foreground then press CONTROL-C (this will kill that running process). Next give the command
-

```
            jobs -l  (note the process IDs are also given,
you could also use ps to get the process IDs)
```

Kill the remaining 2 processes one at a time with
the *kill* command you learned about in assignment 9 (chapter 7).

Kill one using the % along with the job number and the other using the process ID (I'm just trying to get you to use different options).

- ***Do a window grab of the terminal window and include it in your assignment 10 document.***



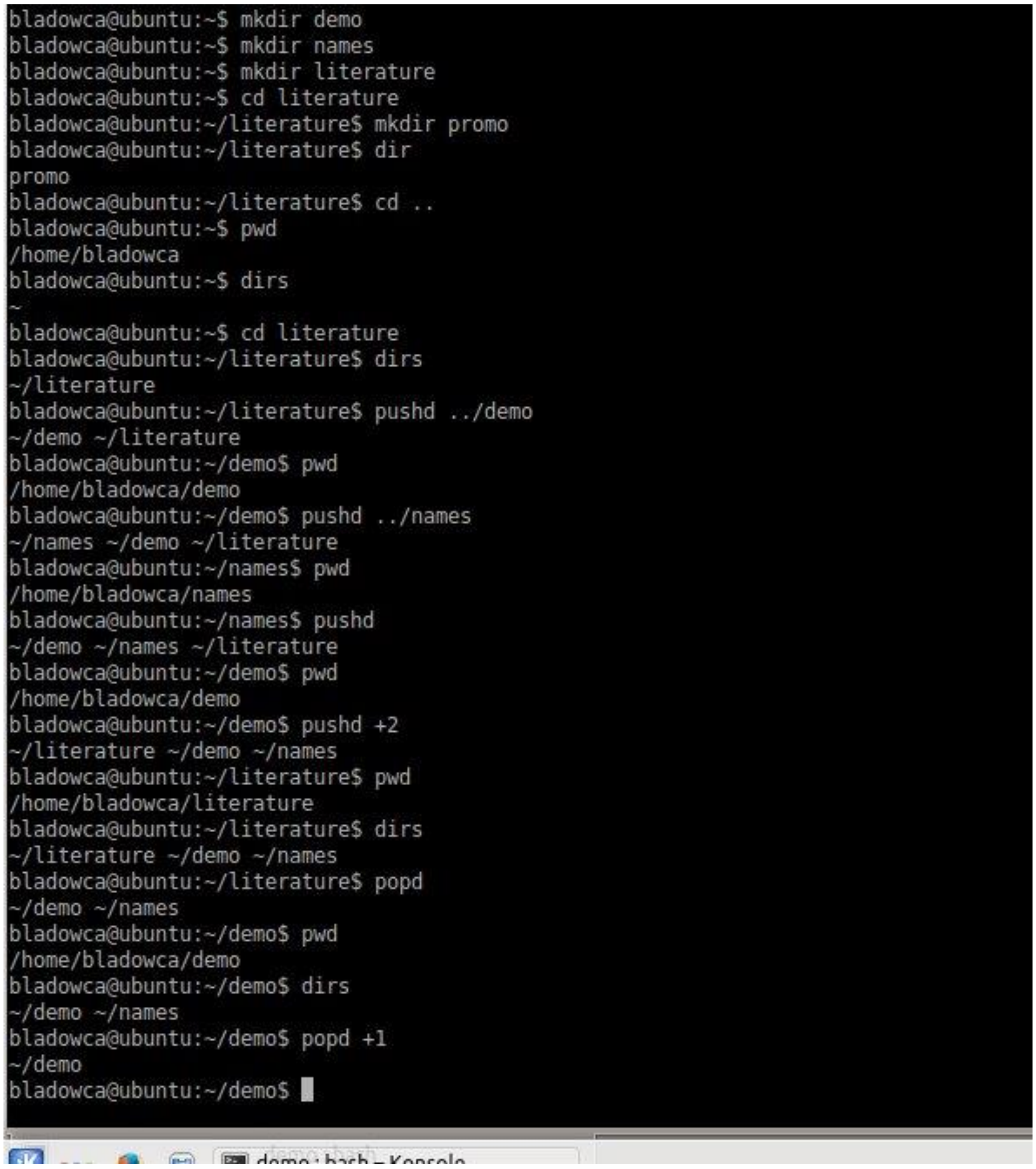
bladowca : bash – Konsole

File Edit View Bookmarks Settings Help

```
bladowca@ubuntu:~$ jobs -l
[2]   7887 Running                 xclock -digital -update 1 &
[3]   7888 Running                 xeyes &
[4]   7918 Running                 xclock -digital -update 1 &
[5]   7919 Running                 xclock -update 1 &
[6]   7920 Running                 xeyes &
[7]-  7921 Running                 xclock &
[8]+  7928 Stopped                 ( sleep 5; cat > mytext )
bladowca@ubuntu:~$ kill 2
bash: kill: (2) - Operation not permitted
bladowca@ubuntu:~$ kill %2
bladowca@ubuntu:~$ kill 7888
[2]   Terminated              xclock -digital -update 1
bladowca@ubuntu:~$ 
```

- Note that for killing processes that were not started from the terminal window, you would have to use the process ID. You've seen this before. Try the command *ps -x* to see all the jobs.

- This activity will make sure you have an understanding of the directory stack and manipulating it.
  - Open a terminal window and resize it vertically so that it has maximum height from the top to bottom of the screen (more room for me to see your results when you do the window grab).
  - Experiment with manipulating the directory stack as described on pages 310-312. You already have a directory structure that you created for assignment 9.
  - ***Do a window grab of the terminal window and include it in your assignment 10 document.***



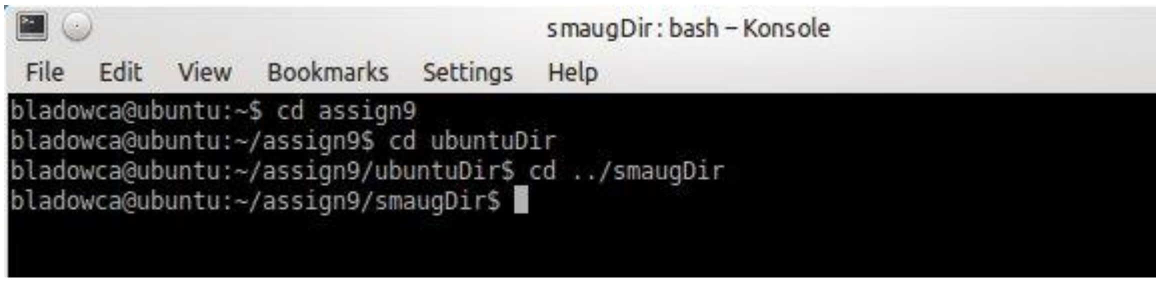
```
bladowca@ubuntu:~$ mkdir demo
bladowca@ubuntu:~$ mkdir names
bladowca@ubuntu:~$ mkdir literature
bladowca@ubuntu:~$ cd literature
bladowca@ubuntu:~/literature$ mkdir promo
bladowca@ubuntu:~/literature$ dir
promo
bladowca@ubuntu:~/literature$ cd ..
bladowca@ubuntu:~$ pwd
/home/bladowca
bladowca@ubuntu:~$ dirs
~
bladowca@ubuntu:~$ cd literature
bladowca@ubuntu:~/literature$ dirs
~/literature
bladowca@ubuntu:~/literature$ pushd ../demo
~/demo ~/literature
bladowca@ubuntu:~/demo$ pwd
/home/bladowca/demo
bladowca@ubuntu:~/demo$ pushd ../names
~/names ~/demo ~/literature
bladowca@ubuntu:~/names$ pwd
/home/bladowca/names
bladowca@ubuntu:~/names$ pushd
~/demo ~/names ~/literature
bladowca@ubuntu:~/demo$ pwd
/home/bladowca/demo
bladowca@ubuntu:~/demo$ pushd +2
~/literature ~/demo ~/names
bladowca@ubuntu:~/literature$ pwd
/home/bladowca/literature
bladowca@ubuntu:~/literature$ dirs
~/literature ~/demo ~/names
bladowca@ubuntu:~/literature$ popd
~/demo ~/names
bladowca@ubuntu:~/demo$ pwd
/home/bladowca/demo
bladowca@ubuntu:~/demo$ dirs
~/demo ~/names
bladowca@ubuntu:~/demo$ popd +1
~/demo
bladowca@ubuntu:~/demo$
```

- ***In your assignment 10 document, describe what the following keyword variables contain.***
  - ***HOME – identifies your home directory***
  - ***PATH – determines which directories the shell searches and in what order to locate commands that you give the shell***

- This activity will make sure you have an understanding of setting and changing the shell prompt.
  - Open a terminal window.
  - From your home directory, give the necessary commands that will change your shell prompt to *[hostname]username/directorypath>* . For example, for me, using the directories created in assignment 9, see below that starts in my home directory (prompt is boldface).

    [brekke360]brekke~> cd assign9
    [brekke360]brekke~/assign9> cd ubuntuDir
    [brekke360]brekke~/assign9/ubuntuDir> cd ../smaugDir
    [brekke360]brekke~/assign9/smaugDir>

  - After you've changed your shell prompt as required, give the commands as above and ***do a window grab of the terminal window and include it in your assignment 10 document.***


smaugDir : bash – Konsole
File Edit View Bookmarks Settings Help

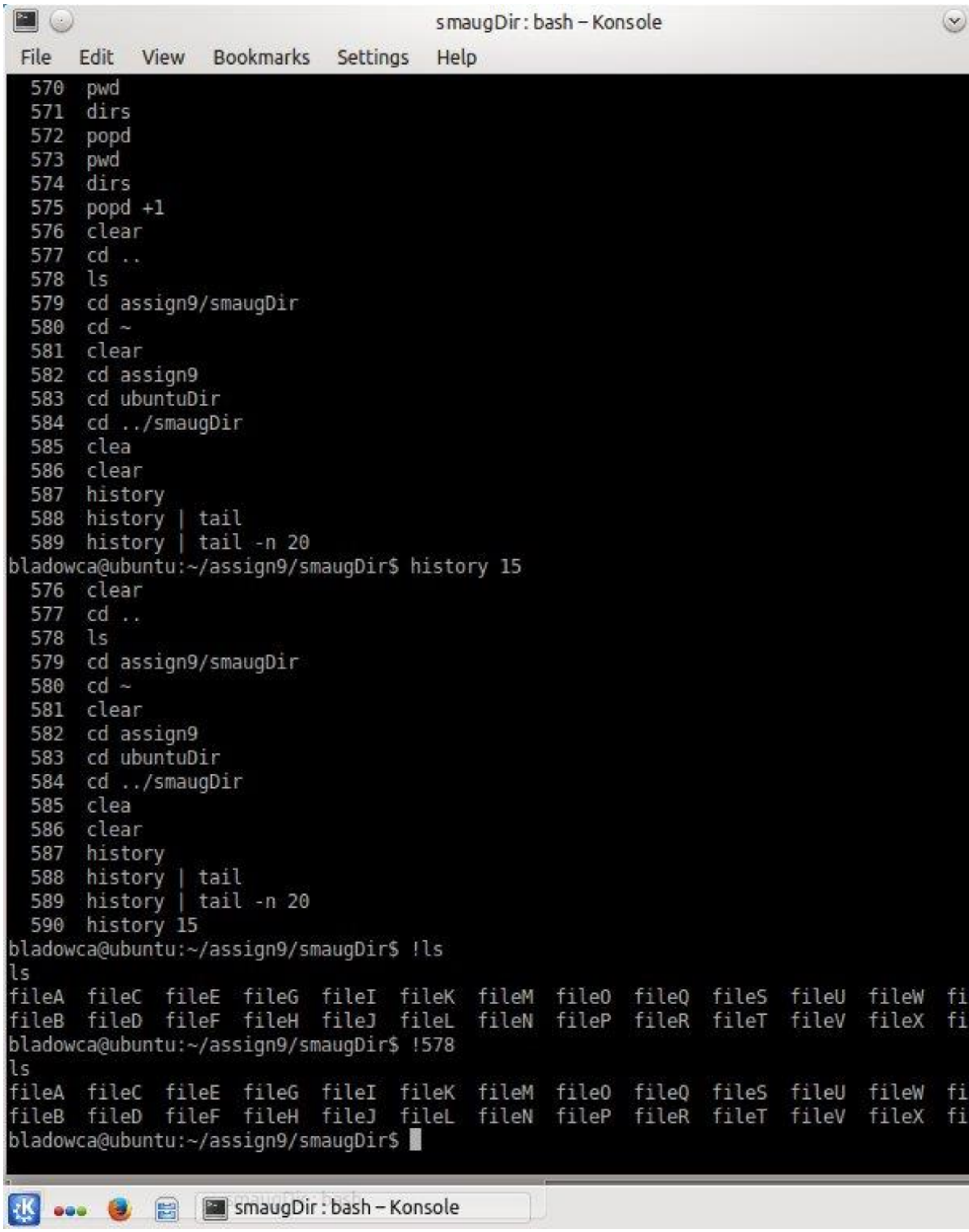
```
bladowca@ubuntu:~$ cd assign9
bladowca@ubuntu:~/assign9$ cd ubuntuDir
bladowca@ubuntu:~/assign9/ubuntuDir$ cd ../smaugDir
bladowca@ubuntu:~/assign9/smaugDir$ 
```

  - Note that if you are changing the prompt on the command line, it will change the prompt for only the current terminal window. If you want a permanent change, you can change it in the file*.bashrc*

- This activity will make sure you have an understanding of the *history* utility and associated options
  - Open a terminal window and resize it vertically so that it has maximum height from the top to bottom of the screen (more room for me to see your results when you do the window grab).
  - Type the following *history* commands. The *history* utility will output your last 1000 commands that you've given. Combining this with your prior knowledge, you should know what the following command produce and it should serve as a review.
  - *`history`*
  - *`history | tail`*

-
  ```
  history | tail -n 20
  history 15
  ```

- The ! (exclamation point) references events. Pick one of the event numbers and type ! followed by the event number from the list to re-execute the command.
- ***Do a window grab of the terminal window and include it in your assignment 10 document.***


smaugDir : bash – Konsole
File Edit View Bookmarks Settings Help

```
  570  pwd
  571  dirs
  572  popd
  573  pwd
  574  dirs
  575  popd +1
  576  clear
  577  cd ..
  578  ls
  579  cd assign9/smaugDir
  580  cd ~
  581  clear
  582  cd assign9
  583  cd ubuntuDir
  584  cd ../smaugDir
  585  clea
  586  clear
  587  history
  588  history | tail
  589  history | tail -n 20
bladowca@ubuntu:~/assign9/smaugDir$ history 15
  576  clear
  577  cd ..
  578  ls
  579  cd assign9/smaugDir
  580  cd ~
  581  clear
  582  cd assign9
  583  cd ubuntuDir
  584  cd ../smaugDir
  585  clea
  586  clear
  587  history
  588  history | tail
  589  history | tail -n 20
  590  history 15
bladowca@ubuntu:~/assign9/smaugDir$ !ls
ls
fileA  fileC  fileE  fileG  fileI  fileK  fileM  fileO  fileQ  fileS  fileU  fileW  fi
fileB  fileD  fileF  fileH  fileJ  fileL  fileN  fileP  fileR  fileT  fileV  fileX  fi
bladowca@ubuntu:~/assign9/smaugDir$ !578
ls
fileA  fileC  fileE  fileG  fileI  fileK  fileM  fileO  fileQ  fileS  fileU  fileW  fi
fileB  fileD  fileF  fileH  fileJ  fileL  fileN  fileP  fileR  fileT  fileV  fileX  fi
bladowca@ubuntu:~/assign9/smaugDir$ 
```

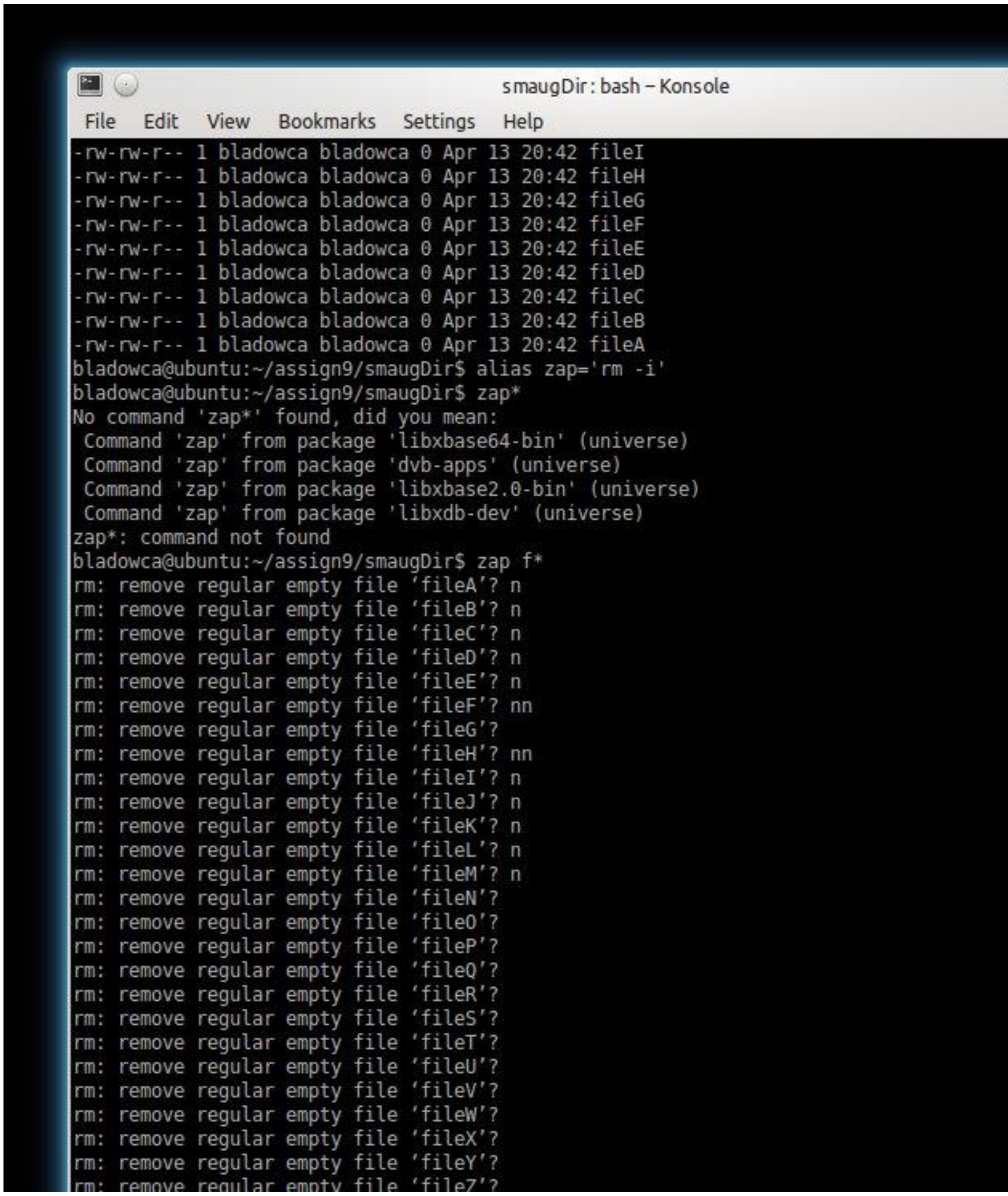
- This activity will make sure you have an understanding of aliases.

- Open a terminal window and resize it vertically so that it has maximum height from the top to bottom of the screen (more room for me to see your results when you do the window grab).
- Type the following commands.
-
-

```
        ls
        ls -F
        alias ls
```

  The second command shows that the *-F* option will add a / following directory names, an * after executables, and an @ after soft links. The third command shows why there are colors for different types of files as the *ls* command has been aliased.

- Type the command *alias* on a line by itself. This will show you all the aliases that have been set by default when your account was created.
- Experiment with some of the example aliases in the *Examples of Aliases* section on pages 348-349.
- ***Do a window grab of the terminal window and include it in your assignment 10 document.***



smaugDir : bash – Konsole
File Edit View Bookmarks Settings Help

```
-rw-rw-r-- 1 bladowca bladowca 0 Apr 13 20:42 fileI
-rw-rw-r-- 1 bladowca bladowca 0 Apr 13 20:42 fileH
-rw-rw-r-- 1 bladowca bladowca 0 Apr 13 20:42 fileG
-rw-rw-r-- 1 bladowca bladowca 0 Apr 13 20:42 fileF
-rw-rw-r-- 1 bladowca bladowca 0 Apr 13 20:42 fileE
-rw-rw-r-- 1 bladowca bladowca 0 Apr 13 20:42 fileD
-rw-rw-r-- 1 bladowca bladowca 0 Apr 13 20:42 fileC
-rw-rw-r-- 1 bladowca bladowca 0 Apr 13 20:42 fileB
-rw-rw-r-- 1 bladowca bladowca 0 Apr 13 20:42 fileA
bladowca@ubuntu:~/assign9/smaugDir$ alias zap='rm -i'
bladowca@ubuntu:~/assign9/smaugDir$ zap*
No command 'zap*' found, did you mean:
 Command 'zap' from package 'libxbase64-bin' (universe)
 Command 'zap' from package 'dvb-apps' (universe)
 Command 'zap' from package 'libxbase2.0-bin' (universe)
 Command 'zap' from package 'libxdb-dev' (universe)
zap*: command not found
bladowca@ubuntu:~/assign9/smaugDir$ zap f*
rm: remove regular empty file ‘fileA’? n
rm: remove regular empty file ‘fileB’? n
rm: remove regular empty file ‘fileC’? n
rm: remove regular empty file ‘fileD’? n
rm: remove regular empty file ‘fileE’? n
rm: remove regular empty file ‘fileF’? nn
rm: remove regular empty file ‘fileG’?
rm: remove regular empty file ‘fileH’? nn
rm: remove regular empty file ‘fileI’? n
rm: remove regular empty file ‘fileJ’? n
rm: remove regular empty file ‘fileK’? n
rm: remove regular empty file ‘fileL’? n
rm: remove regular empty file ‘fileM’? n
rm: remove regular empty file ‘fileN’?
rm: remove regular empty file ‘fileO’?
rm: remove regular empty file ‘fileP’?
rm: remove regular empty file ‘fileQ’?
rm: remove regular empty file ‘fileR’?
rm: remove regular empty file ‘fileS’?
rm: remove regular empty file ‘fileT’?
rm: remove regular empty file ‘fileU’?
rm: remove regular empty file ‘fileV’?
rm: remove regular empty file ‘fileW’?
rm: remove regular empty file ‘fileX’?
rm: remove regular empty file ‘fileY’?
rm: remove regular empty file ‘fileZ’?
```

- Note that if you created aliases from the command line, once you close your terminal window, they are gone. You can make them permanent by adding them to the bottom of the *.bashrc*file. You can remove an alias with the *unalias* command.

4. ***What did you think of this assignment? How long did it take you? Do you have any suggestions for the next time I teach this class?***

   Only 1 left. Getting tough to not half ass them when they start taking too much time... Senioritis may play a part in that too. Approximately 4+ hours on this assignment.

5. ***What did you learn from this assignment?***

   All of it was new. I wasn’t aware of different shells or grouping scripts.