Advanced Shell Usage III.A

- History
  - Variables that Control the History Mechanism
  - Using the History Mechanism
  - Using `fc` to Edit and Run an Old Command
- The Readline Library
  - Readline Completion
  - Pathname Completion
  - Command Completion
  - Variable Completion
- Aliases
  - Single Quotes Versus Double Quotes in Aliases
  - Examples of Aliases
- Functions
- Where to Define Variables, Aliases and Functions

History

- If you hit ↑ at the command line, the shell will bring back your *last* command
- When you do this, you are using the *history mechanism*
  - The history mechanism maintains a list of the commands you have run
  - These command line entries are called **events**
  - Each time you hit the up arrow, the history mechanism shows you a previous command line

History

- By repeatedly hitting ↑ , you can go back in time to see previous commands
- If you go too far, hit the down arrow key ↓ to go *forward* in time

- The history list also serves as a *record* of what you have done.
  - If a command does not work, you can use this history to see what you did wrong
  - To view the history list, use the ***`history`*** command…

```
$ history
    2  exit
    3  cd
    4  cd it244/work
    5  pwd
    6  rm -rf *
    7  cd ~/it244/work
    8  pwd
    9  cp ~ghoffmn/examples_it244/bother.sh .
   10  ls /home/ghoffmn/examples_it244
   11  cp ~ghoffmn/examples_it244/bother.sh .
   12  ./bother.sh
   13  ./bother.sh &
   14  jobs
   ...
```

History

- If you run ***`history`*** without an argument, it will display *<u>all</u>* the events in this history list
- By default, this list contains 500 values, which is probably more than you want to see!
- To show a smaller number of events, run ***`history`*** **,** followed by a number

```
$ history 10
  498  ps
  499  exit
  500  exit
  501  history
  502  cd
  503  cd it244
  504  cd work
  505  ls
  506  history
  507  history 10
```

- Notice that there is no **-** in front of the number, as there **must** be when using ***`head`*** or ***`tail`***

History

- You can also use ***history*** with a pipeline.
- ***Examples:***

```
history | tail -25
history | head -30
history | head -300 | tail -100
history | less
history | tail -250 | less
```

Variables that Control the History Mechanism

- There are three variables that Bash uses to manage the history mechanism:

File	Contents
`HISTFILE`	The location of the file that records the command history. The default is **`~/.bash_history`**
`HISTSIZE`	The maximum number of command lines saved in a list in RAM during a given session
`HISTFILESIZE`	The maximum number of command lines saved in the file specified by **`HISTFILE`** after you quit

- All these variables are **keyword variables**. (Notice that they are all *capitalized*.)

Variables that Control the History Mechanism

- When Unix is set up, these variables are assigned values

```
$ echo HISTFILE: $HISTFILE; echo HISTSIZE: $HISTSIZE;
  echo HISTFILESIZE: $HISTFILESIZE
HISTFILE: /home/it244gh/.bash_history
HISTSIZE: 500
HISTFILESIZE: 500
```

- You can *change the values* of these variables in your `.bash_profile` file

- Your history list is kept in `.bash_history` in your home directory – unless you change `HISTFILE`

Variables that Control the History Mechanism

- So the history mechanism uses **two** lists:

List	Location	Size
File list	`~/.bash_history`	`$HISTFILESIZE`
Memory list	`RAM`	`$HISTSIZE`

- `~/.bash_history` contains commands from your last terminal session
- Commands from your current terminal session are stored in a separate list

Variables that Control the History Mechanism

- When you first log in, the two lists are identical because the *initial* value of the list in memory is taken from the contents of `~/.bash_history`
- As you enter new commands at the terminal…
    - these commands are *added* to the end of the list in RAM, and…
    - older commands are *removed* to keep the size of the RAM list limited to the value of `HISTFILESIZE`

Variables that Control the History Mechanism

- When you *quit* your terminal session, the contents of the RAM list are added to **`~/.bash_history`**
- If **`.bash_history`** already has the maximum number of lines, events from the top of the list are *deleted* to make room for the new entries

Using the History Mechanism

- The ↑ and ↓ keys are not the only way to use the history list
- If you only had the arrow keys to get back an old command, it would be very annoying retrieving a very old event
- But, the history mechanism provides an easier way!
- If you know the event number (which you can get by running ***`history`***)…

Using the History Mechanism

```
$ history 5
  515  cd
     ~ghoffmn/examples_it244/
  516  pwd
  517  echo $PATH
  518  cd
  519  history 5
```

- …then you can run the command again by using an exclamation mark `!`, followed by the event number

```
$ !517
echo $PATH

/usr/local/sbin:/usr/local/
bin:/usr/sbin:/usr/bin:/sbin:
/bin:/usr/games
```

- Notice that the history mechanism *prints out* the old command line before running it

Using the History Mechanism

- There must be **no space** between the **!** and the number, or you will get an *error message*

```
$ ! 517
517: command not found
```

- If you follow the **!** with a letter the last command line that began with *that letter* will be run

```
$ !e
echo $PATH

/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/
bin:/sbin:/bin:/usr/games
```

Using *fc* to Edit and Run an Old Command

- The utility ***fc*** (**f**ix **c**ommand) allows you to edit a previous command line – and then *re-execute* it.
- ***fc*** is a built-in, so it executes quickly
- When run with *no* arguments, ***fc*** will bring up an editor window, holding the last command line
- You can then
  - Modify the command in the editor window
  - Save your changes
  - Execute the modified command

Using *fc* to Edit and Run an Old Command

- Running ***fc*** with an *event number* will put that command in the editor window
- If you change your mind while in ***fc*** editor, you must *delete all text* from the window
- If you *don't*, then ***fc*** will try to execute whatever you have left in the window
- ***fc*** can also be used to view the history list

Using *fc* to Edit and Run an Old Command

- When run with the `-l` option, ***fc*** will list the *last 16* command lines:

```
$ fc -l
511   ls
512   cd
513   ls
514   history 5
515   cd
  ~ghoffmn/examples_it244/
516   pwd
...
...
517   echo $PATH
518   cd
519   history 5
520   echo $PATH
521   ! 517
522   echo $PATH
523   traceroute -a
standford.edu
524   echo $PATH
525   echo $PATH
526   echo $PATH
```

Using *fc* to Edit and Run an Old Command

- You can also tell ***fc*** to list all command lines starting with a certain event number
- You do this by running ***fc -l***, followed by a space and a number

```
$ fc -l 522
522   echo $PATH
523   traceroute -a standford.edu
524   echo $PATH
525   echo $PATH
526   echo $PATH
527   fc -l
```

Using *fc* to Edit and Run an Old Command

- You can also have ***fc -l*** list a range of events
- To do this, follow ***fc -l*** with two numbers

```
$ fc -l 522 525
522  echo $PATH
523  traceroute -a standford.edu
524  echo $PATH
525  echo $PATH
```

- If you run ***fc -l*** with two strings, it will list a range of command lines.

Showing a Range with *fc*

- The list will **start** with the last command line that matches the first string and **end** with the last command that matches the last string

```
$ history 10
   521   ! 517
   522   echo $PATH
   523   traceroute -a
standford.edu
   524   echo $PATH
   525   echo $PATH
   526   echo $PATH
   ...
```

```
...
   527   fc -l
   528   fc -l 521
   529   fc -l 522 525
   530   history 10
```

```
$ fc -l t f
523   traceroute -a
standford.edu
524   echo $PATH
525   echo $PATH
526   echo $PATH
527   fc -l
528   fc -l 521
529   fc -l 522 525
```