Outline

- Shell variables and customization

- To be done by the next class
  - Read the sections on prompt, shell variables, and environment variables of the manual pages of tcsh given by “man tcsh” (optional, not required for exams)
Systems Steps When You Login In

- When you login into shell.cs.fsu.edu, the UNIX does the following for you
  - It starts the shell program that is specified in /etc/passwd for you
  - The following is my entry in /etc/passwd
    liux:x:23456:300:Xiuwen Liu:/home/faculty/liux:/etc/reg-tcsh
  - While most of us run the same tcsh program, some commands work in my account may not work in yours
Systems Steps When You Login In

- When you login into shell.cs.fsu.edu, the UNIX does the following for you
  - It starts the shell program that is specified in /etc/passwd for you
  - The following is my entry in /etc/passwd:
    liux:x:23456:300:Xiuwen Liu:/home/faculty/liux:/etc/reg-tcsh
  - While most of us run the same tcsh program, some commands work in my account may not work in yours
    - Because an account can be customized through shell and environment variables
Shell Variables

- For example, you can set the command prompt in a particular way
- These and other features are achieved by setting/changing shell variables
- Shell variables are the variables used by shell and other programs
  - For example, history specifies how many command entries to be used
Environment Variables

- There are a set of special variables available to shell, called environment variables
  - They can also be set or changed
  - But these environment variables will be passed to the new shell or programs
  - In contrast, regular shell variables will NOT be passed to the new shell or programs
Common Shell Variables

- When a shell is started, it will create a set of predefined shell/environment variables
  - PWD/cwd
    - The current working directory
  - USER/user
    - The current user name
  - GROUP/group
    - The group the current user belongs to
  - DISPLAY
    - Used for X-windows (which we will cover later)
Important Shell Variables

- PATH and path
  - PATH is an environment variable and path is a shell variable
    - By design, the directories in them are kept the same
  - When we run a program by typing its name only, the shell/UNIX will search the command under the directories listed in PATH
  - If this variable is not set properly, some UNIX commands will not be available to you (unless you specify the absolute path to of the commands)
How to Change Shell Variables

- In tcsh, use “set” to change or create a shell variable
  - set name=value
  - set name
    - It will define a shell variable with the given name but the value of the variable is empty
  - set by itself will show all the currently defined shell variables
- To delete a defined shell variable, use “unset”
- Note that set and unset are built-in shell commands
  - In other words, there is no corresponding external program
Shell Variable Examples

- `set history=512`
  - It will change the number of history entries to 512

- `set autolist`
  - If set, all matching commands will be listed if auto-completion is ambiguous

- `set noclobber`
  - If set, a file used for output redirection must NOT exist; otherwise, the shell generates an error stating that the file exists already
    - This helps preventing accidental destruction of files
How to Change/Create Environment Variables

- In tcsh, we can create/change environment variables using `setenv`
  - `setenv name value`
    - Create/update the environment variable name to the given value
  - `setenv name`
    - Create/update the environment variable name to an empty string
  - `setenv by itself will show all the defined environment variables`
- To delete an environment variable, use "unsetenv"
Potential Exam Questions

- True/false
  - In tcsh, a shell variable with an empty value is the same as the shell variable is not defined. [T/F]
  - In tcsh, a user can define additional environment variables but not additional shell variables. [T/F]
  - Changing some shell variables can cause UNIX commands to stop working. [T/F]
  - After executing “unset path”, the following will not list the files in your home directory. [T/F]
    - `cd ~`
    - `ls`
Potential Exam Questions

- **True/false**
  - In tcsh, a shell variable with an empty value is the same as the shell variable is not defined. [F]
  - In tcsh, a user can define additional environment variables but not additional shell variables. [F]
  - Changing some shell variables can cause UNIX commands to stop working. [T]
  - After executing “unset path”, the following will NOT list the files in your home directory. [T]
    ```
    cd ~
    ls
    ```
Potential Exam Questions

- What would be the output after executing the following commands?
  ```
  setenv MYCOP3353 /home/faculty/liux/COP3353
  tcsh
  setenv | grep MYCOP3353
  ```

- What would be the output after executing the following commands?
  ```
  set mycop3353 = /home/faculty/liux/COP3353
  tcsh
  set |grep mycop3353
  ```
Potential Exam Questions

- What would be the output after executing the following commands?
  ```
  setenv MYCOP3353 /home/faculty/liux/COP3353
tcsh
setenv |grep MYCOP3353
  ```
  - `MYCOP3353=/home/faculty/liux/COP3353`

- What would be the output after executing the following commands?
  ```
  set mycop3353 = /home/faculty/liux/COP3353
tcsh
setenv |grep mycop3353
  ```
  - Nothing
Potential Exam Questions

What would be the output after executing the following commands?

```plaintext
set mycop3353 = /home/faculty/liux/COP3353
tcsh
set | grep mycop3353
exit
set | grep mycop3353
```
Potential Exam Questions

- What would be the output after executing the following commands?
  
  ```
  set mycop3353 = /home/faculty/liux/COP3353
tcsh
set | grep mycop3353
exit
set | grep mycop3353
  ```

- `mycop3353 /home/faculty/liux/COP3353`
Potential Exam Questions

- Will the files in your home directory be listed after executing the following commands? Justify your answer.
  
  ```
  set path /bin/tcsh
  cd ~
  ls
  ```
Potential Exam Questions

- Will the files in your home directory be listed after executing the following commands? Justify your answer.
  
  set path
  /bin/tcsh
  cd ~
  ls

- Yes. While shell variable path was changed to an empty string, the changed path was not passed to the new shell program and therefore ls will work.
Potential Exam Questions

- Will the files in your home directory be listed after executing the following commands?
  
  ```
  set path /bin/tcsh
cd ~
exit
cd ~
lis
  ```
Potential Exam Questions

- Will the files in your home directory be listed after executing the following commands?
  
  set path
  /bin/tcsh
  cd ~
  exit
  cd ~
  ls

- No.
Ways to Customize Your Account

- When a shell is started, it will execute the commands in two dot files under your home directory
  - It will first execute all the commands in .tcshrc
  - It will then execute all the commands in .login
  - It will also execute system source files /etc/csh.cshrc and /etc/csh.login (but users cannot change them)
- You can include UNIX commands and shell variables/environment variables to customize your account in these files
Customizing Prompt

- tcsh provides ways for you to “program” your prompt
  - Examples
    - `set prompt = “%#”`
    - `set prompt = “$USER ”`
    - `set prompt = "$USER [%h] “`
    - `set prompt = “liux@shell.cs.fsu.edu:%c3>”`
    - `set prompt = "%m [%h] %B[@]%b [%/] “`
Formatting Sequences for Prompt

%/ The current working directory.
%~ The current working directory, but with one’s home directory represented by ‘~’ and other users’ home directories represented by ‘~user’ as per Filename substitution. ‘~user’ substitution happens only if the shell has already used ‘~user’ in a pathname in the current session.
%c[[0]n], %.[[0]n]
The trailing component of the current working directory, or n trailing components if a digit n is given. If n begins with ‘0’, the number of skipped components precede the trailing component(s) in the format ‘/<skipped>trailing’. If the ellipsis shell variable is set, skipped components are represented by an ellipsis so the whole becomes ‘...trailing’. ‘~’ substitution is done as in ‘%-’ above, but the ‘~’ component is ignored when counting trailing components.
%c Like %c, but without ‘~’ substitution.
%h, %!, !
The current history event number.
%M The full hostname.
%m The hostname up to the first ‘.’.
%S (%s)
Start (stop) standout mode.
Formatting Sequences for Prompt

%b (%%b)  
Start (stop) boldfacing mode.

%u (%%u)  
Start (stop) underline mode.

%t, %@  
The time of day in 12-hour AM/PM format.

%T  
Like '%t', but in 24-hour format (but see the `ampm` shell variable).

%p  
The 'precise' time of day in 12-hour AM/PM format, with seconds.

%P  
Like '%p', but in 24-hour format (but see the `ampm` shell variable).

%c  
c is parsed as in `bindkey`.

%c  
c is parsed as in `bindkey`.

%%  
A single '%'.

%n  
The user name.

%j  
The number of jobs.

%d  
The weekday in 'Day' format.

%D  
The day in 'dd' format.

%m  
The month in 'Mon' format.

%W  
The month in 'mm' format.

%y  
The year in 'yy' format.

%Y  
The year in 'yyyy' format.

%l  
The shell’s tty.
Formatting Sequences for Prompt

\%L  Clears from the end of the prompt to end of the display or the end of the line.
\%S  Expands the shell or environment variable name immediately after the '\$'.
\%#  '=>' (or the first character of the promptchars shell variable) for normal users, '#' (or the second character of promptchars) for the superuser.
\%{string%}  Includes string as a literal escape sequence. It should be used only to change terminal attributes and should not move the cursor location. This cannot be the last sequence in prompt.
\%?  The return code of the command executed just before the prompt.
\%R  In prompt2, the status of the parser. In prompt3, the corrected string. In history, the history string.

'\%B', '\%S', '\%U' and '\%{string%}' are available in only eight-bit-clean shells; see the version shell variable.

If '\%t', '\%@', '\%T', '\%p', or '\%P' is used, and noding is not set, then print 'DING!' on the change of hour (i.e., ':00' minutes) instead of the actual time.

Set by default to '\%#' in interactive shells.
Creating Aliases

- You can use aliases for commands that you use very often
  - alias by itself shows all the command aliases
  - The following will create an alias for dir, h, and l.
    
    alias dir ls –l
    alias h history
    alias l. ls -d .* --color=TTY

- You can delete an alias using unalias
  - unalias l. will delete .l alias
With shell variables and variable substitutions, one can write programs to perform more complicated tasks than one-line commands.

- Note that .tcshrc and .login are two shell programs.
- We will cover shell programming after the spring break.
You can also customize some aspects of UNIX

- For example, umask can be used to have certain file permissions masked out

- umask value
  - Any bits set in the value will not be set in newly created file/directory permission
    - umask 077
      - All files created will not have any permission bits set for the group or the other
    - umask 000
      - No permission bits will be masked out

- umask
  - Shows the current umask value
Potential Exam Questions

- Suppose that the permission of a new file “file1” created using “touch file1” is “rw-rw-rw-”, where the value returned from umask is 000
  - What would the permission of file1 if umask were 011?
  - What would the permission of file1 if umask were 123?
  - What would the permission of file1 if umask were 777?
Potential Exam Questions

- Suppose that the permission of a new file “file1” created using “touch file1” is “rw-rw-rw-”, where the value returned from umask is 000
  - What would the permission of file1 if umask were 011?
    - rw-rw-rw-
  - What would the permission of file1 if umask were 123?
    - rw-r--r--
  - What would the permission of file1 if umask were 777?
Potential Exam Questions

- True/false
  - After a user executes “umask 777” but before another umask command, all the newly created files by the user will have permission 000. [T/F]
  - On a web server, a umask of 022 should be used so that files created can only be read by others (to avoid security issues). [T/F]
Potential Exam Questions

- True/false
  - After a user executes “umask 777” but before another umask command, all the newly created files by the user will have permission 000. [T]
  - On a web server, a umask of 022 should be used so that files created can only be read by others (to avoid security issues). [T]
All the predefined shell variables, environment variables, formatting sequences and other information in tcsh are available on the manual pages of tcsh
- Using “man tcsh”

While most shells have similar features, details and syntax of them vary from shell to shell
Similar to shell customization, programs often use dot files to allow users to customize their choices.

- For example, pine uses `.pinerc`.
- UNIX email uses `.forward` to allow users to forward their email messages to other accounts.
  - This is my `.forward` file:
    
    wickus@gmail.com
Account newacct

- Will NOT be on exams
- When you created your computer science account, you were asked to ssh to shell.cs.fsu.edu as newacct
  - How is this done?
Account newacct

- Will NOT be on exams
- When you created your computer science account, you were asked to ssh to shell.cs.fsu.edu as newacct
  - How is this done?
  - Here is the entry in /etc/passwd for newacct

newacct:x:997:11:New_Account_Generator:/home/system/newacct:/usr/local/bin/whereami
Summary

- Process management

- To be done by the next class
  - Read the sections on prompt, shell variables, and environment variables of the manual pages of tcsh by “man tcsh” (optional, not required for exams)