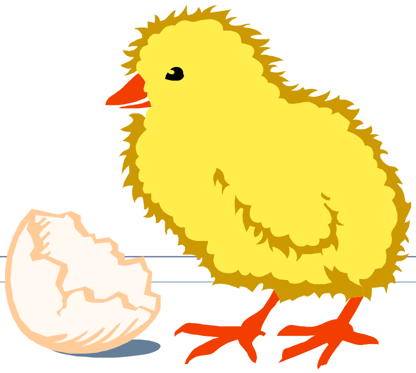
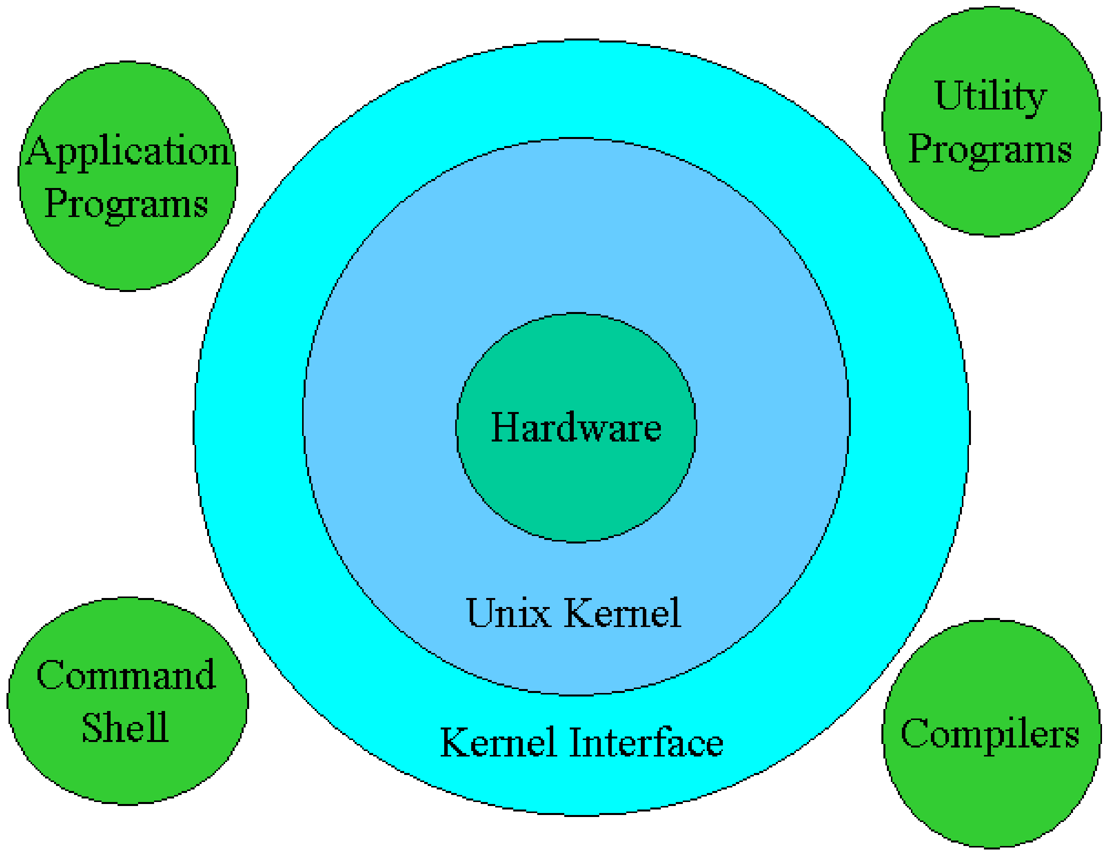
COMP 2021

Unix and Script Programming



Unix Shell

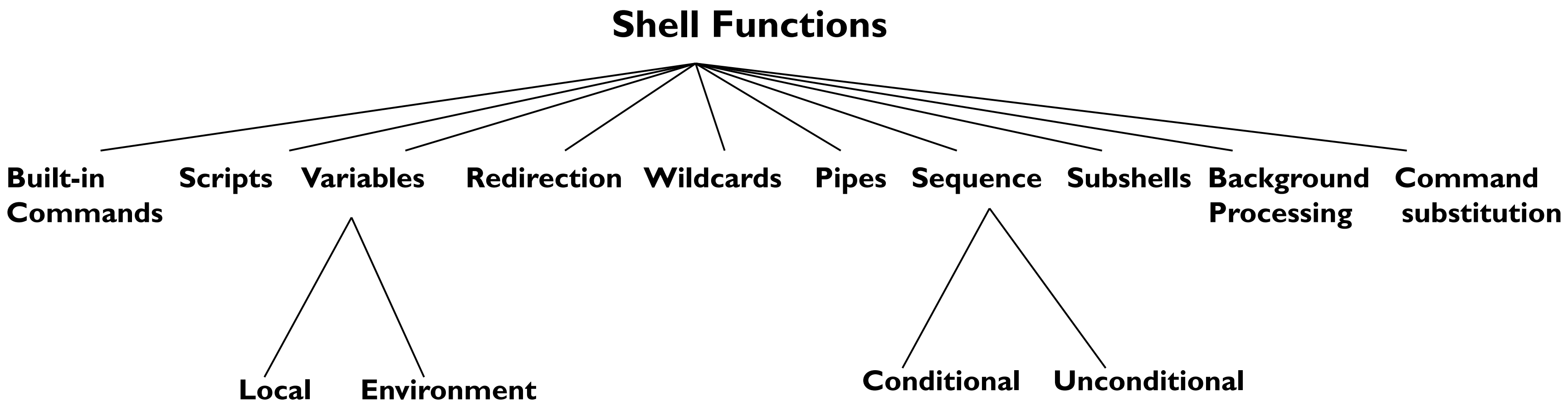
Unix Architecture



Unix Architecture

Unix Shell

- A shell is a program that is an interface between a user and the raw operating system.
  - provide a "command line" interface which allows the user to enter commands which are translated by the shell into some thing the kernel can comprehend and then is sent off to the kernel for it to act upon.



Flavors of Unix Shells

- Two main flavors of Unix Shells
  - Bourne shell (or Standard Shell): sh, ksh, bash, zsh
    - Fast
    - `$` for command prompt
  - C shell : csh, tcsh
    - better for user customization and scripting
    - `%`, `>` for command prompt

- When you are provided with a UNIX account, the system administrator chooses a shell for you.

```
$ echo $SHELL
/bin/tcsh
```

Switching Shells

- You can switch shells by just typing its name:

```
csl2wk09:lixin:119> ps
  PID TTY          TIME CMD
 9806 pts/1    00:00:00 tcsh
10059 pts/1    00:00:00 ps
csl2wk09:lixin:120> sh
sh-3.2$ ps
  PID TTY          TIME CMD
 9806 pts/1    00:00:00 tcsh
10061 pts/1    00:00:00 sh
10062 pts/1    00:00:00 ps
sh-3.2$ tcsh
csl2wk09:lixin:121>
```

Shell Operations

When a shell is invoked, either automatically during a login or manually from a keyboard or script, it follows a preset sequence:

1. It reads a special startup file, typically located in the user's home directory, that contains some initialization information.
   - Each shell's startup sequence is different.

2. It displays a prompt and waits for a user command.

3. If the user enters a Control-D character on a line of its own, this command is interpreted by the shell as meaning "end of input", and it causes the shell to terminate;

otherwise, the shell executes the user's command and returns to step 2.

Class Activity

- Quick go through the contents of following files in tcsh

Startup (in this order):

`/etc/csh.cshrc` (always)

`/etc/csh.login` (login shells)

`.tcshrc` (always)

`.cshrc` (if no .tcshrc file is present)

`.login` (login shells)

Upon termination:

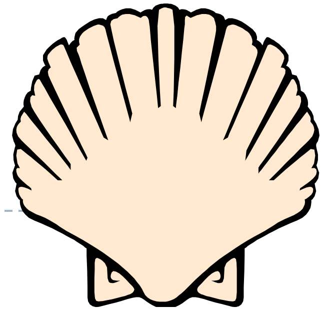
`.logout` (login shells)

Others:

`.history` (saves history based on "`$savehist`")

`.cshdirs` (saves directory stack)

Basic Shell Syntax



command [-[options]] [arg] [arg] ...

- The name of the command is first
- Options are normally single letters that turn an option on or off. They can be combined or given separately.

```
      $ ls -dil                      $ ls -l -d -i
```

- Options sometimes also take a value.

```
      $ head -c 12 file
```

- Commands range from simple utility invocations like:

```
      $ ls
```

- to complex-looking pipeline sequences like:

```
      $ ps -ef  | sort  | ul  -tdumb  | lp
```

- Shell allows you to spread a command through multiple lines with \ character

```
   $ date; sort names; \
      who
```

Utilities and Built-in Commands

- Most UNIX commands invoke utility programs that are stored in the directory hierarchy.
  - Utilities are stored in files that have execute permission.
- The built-in commands are coded inside the shell
  - `echo, cd`

```
$ which echo
echo: shell built-in command.
$ which cat
/bin/cat
$ which grep
/bin/grep
$ which ls
ls:      aliased to ls --color=tty
```

Makes directories blue,
executables green,
and soft links aqua

Alias

- The C Shell has the `alias` command, which allows you to create command shortcuts.

```
$ alias ls "ls -F"
$ alias rm "rm -r"
$ alias + "chmod u+x *"
$ alias - "chmod u-x *"
$ alias 2021 "cd /homes/cindy/comp2021"
$ pwd
/bin
$ 2021
$ pwd
/homes/cindy/2021
$ unalias ls
```

- On most Unix machines (except **Mandriva/Mandrake** in CSLab2), if you put the alias commands in your `.cshrc` file, you can use them every time you login.

Metacharacters

- Some characters are processed specially by a shell and are known as metacharacters.

Symbol	Meaning
>	Output redirection; writes standard output to a file.
>>	Output redirection; appends standard output to a file.
<	Input redirection; reads standard input from a file.
*	File-substitution wildcard; matches zero or more characters.
?	File-substitution wildcard; matches any single character.
[...]	File-substitution wildcard; matches any character between the brackets.
` `	Command substitution

Metacharacters (cont.)

Symbol	Meaning
|	Pipe symbol; sends the output of one process to the input of another.
;	Used to sequence commands.
||	Conditional execution; executes a command if the previous one fails.
&&	executes a command if the previous one succeeds.
&	Runs a command in the background.
#	All characters that follow up to a new line are ignored by the shell and program(i.e. used for a comment)
$	Expands the value of a variable.
( )	Groups commands

- To escape the special meaning of a metacharacter, precede it by a backslash(\) character.

Command Options

- Double quotes and single quotes are a bit different. For now, you can use them interchangeably.

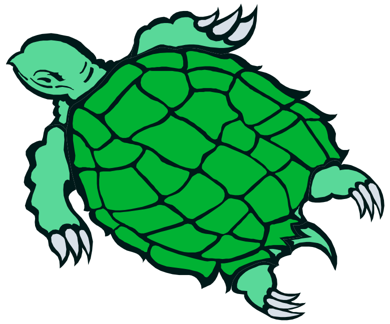
```
$ grep ‘is fun’ comp2021
 comp2021 is fun
```

- To escape a single character (prevent it from being treated specially) proceed it with a backslash:

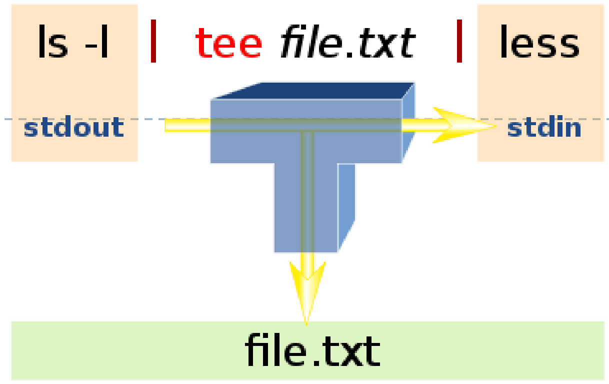
```
$ grep ”We\’ll” letter2

by my office. We'll tidy up a few more things before

$ echo ”*”
*
$ echo ‘*’
*
$ echo \*
*
$ echo *
file1  file2  file3  file4
```



Combining Commands



- Pipes
- Tee
- Parentheses
  - To create a "single command" out of a group of commands (especially useful before a pipe):

    ```
    $ (cat file1; cat file2) | sort > list
    ```

  - To run a set of commands in their own subshell (especially when trying to limit the effect of a `cd` command):

    ```
    $ (cd sub; ls | wc -l); ls | wc -l
            3
           25
    ```

    This line has the effect of counting the files in `sub`, and then counting the files in the current directory.

Filename Substitution (Wildcards)

- All shells support a wildcard facility that allows you to select files that satisfy a particular name pattern from the file system

Wildcard	Meaning
*	Matches any string, including the empty string
?	Matches any single character
[…]	Matches any one of the characters between the brackets. A range of characters may be specified by separating a pair of characters by a hyphen.

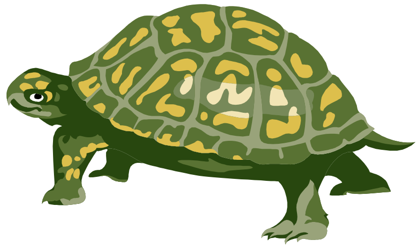
Pattern Matching Examples

- The notation “[abcd]” matches any single one of the enclosed characters.

```
$ ls [il]*
it          it1         ith             letter1          letter4
```

- The notation “[a-z]” matches any lowercase letter.
- The notation “[0-9]” matches any digit character.
- The notation “[0-59]” matches any the digit characters 0, 1, 2, 3, 4, 5, and 9.

```
$ ls letter*
letter1   letter4
$ ls letter[0-35]
letter1
$ ls letter[0-24]
letter1   letter4
```



- Most shells allow you to give a list of strings in curly brackets, comma separated:

```
$ ls *{1,.sort}
NAMES1          it1             names.sort    s1@
f1              letter1         passwd.sort
```

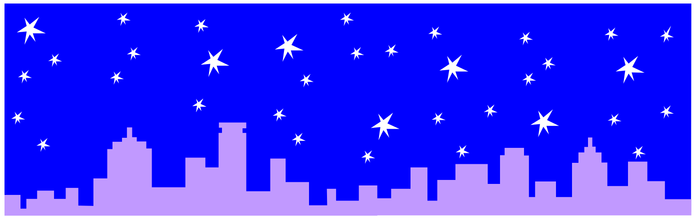
Command Substitution

- A command surrounded by back quote(`) is executed, and its standard output is inserted in the command's place in the entire command line.
  - Any new lines in the output are replaced by spaces.

```
$ echo today is `date`
today is Tue Feb 7 14:51:36 HKT 2014

$ echo there are `who | wc -l` users in the system
there are 5 users in the system
```

Background Processing



- A simple command or pipeline can be put into the background by following it with the `&` character:

```
$ sort 2021StuList > 2021StuList.sort &
[1] 10287
$
[1]    Done  sort 2021StuList > 2021StuList.sort
```

  - The shell will print the process ID (PID), and a *job number* (1, in this case).
  - In some shells, you will be notified when the job is done (you may have to hit return again)

- The background process runs concurrently with the parent shell and does not take control of the keyboard.
- Useful for performing several tasks simultaneously
- (Usually) redirects the output to file instead of terminal

```
     sort 2021StuList | mail lixin &
```