Processes

- Can run several programs simultaneously
  - E.g., a pipeline: each program runs concurrently
- One program can be running in *foreground*
  - E.g., the program reading keyboard input
- Some progs can be running in *background*
  - E.g., long-running programs whose output is collected into a file
- Some programs can be *suspended*
  - Stopped temporarily, for some reason
- Each simultaneously running program is called a *process* or a *job*

Starting in background

- Can start a program in background by appending & to its command line
  - find . –name '*.java' \
    –exec grep foo {} \; \
    –print > filesContainingFoo.txt &
  - <do something interactive>
  - <then go check on output of find>

Controlling processes

- Can suspend the current foreground job by typing ^z (control-z)
- bg
  - moves the current suspended job to background
- fg
  - moves the current background or suspended job to foreground
- jobs
  - shows the current suspended & background jobs, and their job #'s

Processes and process id's

- ps aux
  - shows all the processes on the machine, and their owners, process id's (pids), etc.
- kill *pid*…
  - Kill one or more processes with the given process id's

Defining your own commands

- 3 ways to define your own commands:
  - Write a new program, compile it, and put the executable somewhere in your path
    - Heavyweight
  - Write a *script*, put it somewhere in your path
    - Lightweight
  - Define an *alias*, e.g. in your .cshrc
    - Flyweight

Aliases

- alias *aliasName command arg*…
  - Defines *aliasName* to be an abbreviation for *command arg*…
  - Whenever type *aliasName aliasArg*… at the shell prompt, replaced with *command arg*… *aliasArg*…
    - Doesn't work in other contexts, e.g. –exec args
- alias ll ls –l
- alias k kill -9

Shell scripts

- Aliases work for one-liners
- For more complex tasks, can write *shell scripts*
- A script is a file containing a sequence of regular Unix shell commands
  - includes control structure commands like if, while, foreach, switch
  - includes argument processing operations
- (.cshrc is just a script run at log-in)

Making a script into a program

- Must start with #!/bin/csh
  - This says that /bin/csh should be used to interpret the rest of the lines
  - Can use other interpreter programs, e.g. /bin/perl, /bin/sh, …
- Must be marked as executable
  - chmod +x *scriptName*
- Must be in a directory in the path

Shell script arguments

- The argv shell variable is set to the list of arguments to the shell
  - $argv expands to the list of arguments
    - $* is a synonym for $argv
- $*var*[*n*] refers to the *n*th element of the *var* list
  - $argv[*n*] is the nth shell argument
    - $*n* is a synonym for $argv[*n*]
- $#*var* refers to the length of the *var* list
  - $#argv is the number of shell arguments
- $0 is the name of the script being run

Foreach command

- foreach *varName* (*arg*…)
  … *body command lines* …
  end
  - sets *varName* to each *arg* in turn
    - arg is often a pattern
  - evaluates *body command lines* for each setting

Examples

- foreach f (*.htm *.html)
  echo "moving $f to www/$f"
  mv $f www
  end
- foreach arg ($*)
  … do something to $arg …
  end

Advanced variable substitution

- Often want to process shell variable bindings (e.g. foreach loop variables)
- Can add qualifiers to extract pieces e.g. of pathnames
- if $*var* == a/b/c.d.e, then
  - head: $*var*:h == a/b
  - tail: $*var*:t == c.d.e
  - root: $*var*:r == a/b/c.d
  - extension: $*var*:e == e
- Can repeat modifiers, e.g. $*var*:h:h == a

Example

- foreach f (*.htm)
  ```
  set g = ${f:r}.html
  echo "fixing extension of $f to $g"
  mv $f $g
  end
  ```
- Note that can uses braces after $ to clearly identify the variable subst. expr.

If command

- if (*expr*) then
  … *commands* …
  else if (*expr2*) then
  … *commands* …
  …
  else
  … *commands* …
  endif
  - zero or more else-if cases
  - optional else case

Test expressions

- String comparisons: ==, !=
- String pattern-matching: =~, !~
- Numeric comparisons & operators, e.g. +, <
- Boolean expressions, e.g. &&, ||, !
- Parenthesized subexprs

- if ("$f" == README || "$f" =~ *.java) …
- if ($#argv < 2) …

File test expressions

- Also can test properties of files
  - -e *fileName*: *fileName* exists?
  - -f *fileName*: *fileName* is a plain file?
  - -d *fileName*: *fileName* is a directory?
  - -x *fileName*: *fileName* is executable?
- if (-e $f && ! –d $f) …

See also

- while
- break, continue
- switch, case, default, breaksw
- shift
- exit
- pushd, popd
- time

Shell as a programming language

- How is shell script programming different from regular programming?
  - Types
  - Declarations
  - Procedures
  - Data structures
  - Primitive/built-in operations
  - Libraries
  - Compilation/execution model