Lecture 4

Redirecting standard I/O,
Appending to files,
and Pipes

COP 3353 Introduction to UNIX

Standard input, output and error

- standard input (0: stdin)
  - The default place where a process reads its input (usually the terminal keyboard)
- standard output (1: stdout)
  - The default place where a process writes its output (typically the terminal display)
- standard error (2: stderr)
  - the default place where a process can send its error messages (typically the terminal display)

Redirecting standard I/O

- Standard input and output can be redirected providing a great deal of flexibility in combining programs and unix tools
- Can redirect standard input from a file using <

```
a.out < input12
```

  - any use of stdin will instead use input12 in this example
- Can redirect standard output to a file using >

```
testprog1 > testout1
```

```
cal > todaycal
```

- Can also redirect stderr and / or stdout at the same time
  - `a.out < input12 > testout`
  - input12 is used as the standard input to a.out and the stdout of a.out is redirected to file testout1

Appending to a file

- If you use the > operator to redirect output to a filename that already exists, any existing data in that file WILL BE OVERWRITTEN!
- How to avoid this? Use >>
- The >> operator *appends* to a file rather than redirecting the output to a file

```
prog1.exe >>assign4
prog2.exe >>assign4
cat endinfo >>assign4
```

Pipes

- Pipes allow the standard output of one program to be used as the standard input of another program
- The pipe operator '|' takes the output from the command on the left and feeds it as standard input to the command at the right of the pipe
- Examples

```
ls | sort -r
cat file.txt | wc -l
```

- Pipes are more efficient as compared to using intermediate files

- Can also use pipes and redirection together.

```
prog1.exe < input.dat | prog2.exe |
prog3.exe > output.dat
```

Redirection & Pipes: The Difference

- Redirection is used between a COMMAND and a FILE. Either redirecting the output of a command to some file, or using some file as input to a command.
  - command > file (output redirection)
  - command < file (input redirection)

- Pipes are used between TWO COMMANDS. You cannot use files with pipes.
  - command1 | command2

  (output of command1 used as input to command2)

Separating commands

- Multiple instructions on one line
  - separate instructions by ‘;’

```
ls -l; cal; date
```

- Suppose you need to continue a command to the next line - use the ‘\’ to do so and then continue your command on the next line

```
cat filename | sort \
| wc
```

```
(These will be more useful when we get
 to shell scripting)
```