Lab 4 - Linux CLI, Shell scripts

Advantages

- Faster, especially for bulk processing
- Low-bandwidth remote access
- Can write scrips

Disadvantages

- Often overly complex
- Doesn't make options easily visible
- Reduced information density

Design Ideas

- Include support for both CLI and GUI
- GUI is wrapper for CLI
- GUI supports launch options that trigger CLI actions
- Launch GUI from CLI?

Shell

- Takes commands from keyboard and passes them to software calls, OS
- Backend for terminal emulators
- Bash is primary shell program

Directories

- `/` - root directory
- `/boot` - kernel and boot files
- `/etc` - system config files
- `/bin` - most system programs
- `/usr/bin` - most user programs
- `/usr` - stuff to support users
- `/var` - files that change as system is running
- `/home` - user's home directories (similar to `C:\users\`)
- `/root` - root's home directory
- `/tmp` - temporary files
- `/dev` - hardware devices
- `/mnt` - common mount location

Navigation

- `pwd` - print working directory (usually starts out as /home/{username})
- `cd` - change directory
  - paths starting with `/` - absolute path
  - paths starting with `.` - relative path
  - paths starting with `..` - paths relative to parent directory
  - paths starting with `~` - paths relative to home directory
  - case sensitive
- ls - list directory contents
  - without arguments - list contents of current directory
  - `ls {directory path}` - list contents of specified directory
  - `ls -l` - verbose list
  - `ls -a` - show hidden files (files that start with `.`)
  - `ls -h` - human readable file sizes
- tab key auto-completes

File Manipulation

- `cp {file} {/path/to/destination}` - copies file to destination
- `mv {old file/dir} {new file/dir}` - moves file from old file to new file (can be used to rename)
- `rm {filename}` or `rm -R {directory}` - removes files or directories. Be **very** careful when using wildcards
- `mkdir {dirname}` - makes directory dirname
- `du` or `du {/path/name}` - disk usage

Files

- `cat {filename}` - print the contents of file
- `{command} | more` - allows scrolling of output of command
- `grep ‘‘expression’’ {/path/name}` - searches files in the given pat for occurrences of “expression”
- `diff {file1} {file2}` - find differences between files

Permissions

- `chmod {permission} {filename}` - sets permission for file.
  - `chmod a+x {filenames}` - add execution permission to files
  - `chmod 777 {filenames}` - 1=can execute, 2=can write, 4=can read. {user}{group}{all}
- `chown` - change file' owner
- `chgrp` - change file's group

I/O Redirection

- `{command1} | {command2}` - pipe output from command1 as input for command2
- `{command} > {filename}` - overwrite the contents of a file with the output of the command
- `{command} >> {filename}` - append the output of a command to a file
- `{command} < {filename}` - provide the contents of a file as input for a command
  - `sort <{file1} >{file2}`

redirection sohould be after other options. order of redirections does not matter.

Shell scripts

```
#!/bin/bash
for file in *
do
    echo $file
done

echo $((5+3))
echo "$USER in $(pwd)"
echo $(cal)
```

Misc

- `{command} &` - runs command in background, freeing up command prompt
- `screen` - virtual CLI window management