System Administration:
Week 6 Notes

March 13, 2006

1 Logs

- Understanding what is in logs helps realize what is up with the operating system: is somebody trying to compromise my server, is a driver crashing my computer, etc
- Can edit file **/etc/syslogd.conf** where logs should go
- **/var/adm/messages**: a catchall file for lots of messages from the Unix kernel and other logging applications like syslogd. Also sometimes used to store miscellaneous log files, including those created by syslog for messages not written to **/usr/adm/messages** or the console.
- **var/adm/lastlog**: stores information about users who are or have logged onto the system. File is in binary and is used by **last**
- **/var/adm/sulog**: records all attempts by users to execute su.
- Unix makes backup files, numbering them sequentially higher
- Files labeled o.logname usually indicate an overflow log. If a log file overflows, all auditing put there stops.
- **/var/adm/utmpx** - keeps information on who is currenly logged in. File used by **who** command.
- **/var/adm/wtmpx** - keeps information on who logs in/out and of when the machine reboots

1.1 Log Rotation

- When running a server, even one at home, the size of logs would become very large. Since the standard is keeping the most recent one without any additional numbers at the end, all of the logs need to be rotated down when the current ones becomes too large or the time is set to rotate it.
- **syslogd** daemon accomplishes this

1.2 Cron Jobs

- It is an automated process which operates at preset intervals. For example, clearing out your accumulated spam at the end of the week. You can also use a daemon to do this, but it would be harder to write and would have to constantly run and take up your system's resrouces.
- Accomplished through **crontab** command, which uses **crond** daemon which constantly runs and check if its time to execute your process
- **crontab -l** lists your current crons
- **crontab -e** edits the file containing your crons
- **contab -r** deletes all of your cron jobs

2 Shell Scripting

- The first line of your shell script will start with #!/bin/bash if you are writing a bash script. You can replace bash by your favorite shell
- Then you would want to have some comments about what the script does, your name, date, etc. Use # for comments which the script will ignore.
- To execute a script you will need to change the permissions to make the file executable with 755 for example and then execute the script with **./**. For example to execute a script foo.sh you will need to type ./foo.sh
- You can reference variables by putting a **$** before them. For example, to display the terminal you are currently using, you would type **echo $TERM**.
- Enclose anything in **" "** that you want treated as a string.
- Use backquotes **‘ ‘** to execute commands. Example: **echo "Today is ‘date‘ "** vs **echo "Today is date."** will print **Today is March 13 22:53:18 PST 2006** vs **Today is date**.
- Use **read** to prompt for input and read it. For example **read fname** will read whatever user inputs and stores it as fname.
- To put more than one command on one line of the shell script, use **;** eg **who; ls**

2.1 If, Else, For, While

- A typical example of how you would use the if
  else statements.

```
if condition
then
if condition
then
.....
..
do this
else
....
..
do this
fi
else
...
.....
do this
fi
```

.

- For for loops you will need to put the regular statement in double parenthesis and a do, for example for ((i=0; i<5; i++)); do echo $i; done

2.2 Wild Card Characters

- * matches any number of characters
- ? matches any single character
- [] matches all characters enclosed in brackets, eg ls [abc]* matches any file names that begin with a b or c. [-] will match a range of characters, for example [1-10] will match any number between 1 through 10.
- ! does not match any of the charcters following it