CS2043 - Unix Tools & Scripting
Lecture 14
Automation

Spring 2015 [1]

Instructor: Nicolas Savva

February 23, 2015

[1] based on slides by Hussam Abu-Libdeh, Bruno Abrahao and David Slater over the years

Announcements

- A4 is out (due 02/28)
- final project timeline
- CMS file checksum

md5sum

Print MD5 (128-bit) checksums

```
md5sum filename(s)
```

md5sum

Check MD5 checksums

```
md5sum -c checksumfile
```

Example

```
md5sum * > myfoldermd5.log
```

We first the checksums of all the files in the folder and store them in the log file.

```
md5sum -c myfoldermd5.log
```

We then check all the checksums in the log file against the files in the folder

diff

```
diff file1 file2
```

Output

- `n{c,a,d}m`: one of line change (c), addition (a), deletion (d) occurred in line `n` of file1 compared to line `m` of file2.
- <: means that this line is exclusive of `file1`
- >: means that this line is exclusive of `file2`

mailx

Example

`mailx [-s "subject"] <e-mail address>`

Send email to the specified address with a particular subject line.
Enters interactive mode to type the body of the letter
(press `Ctrl+D` to finish)
Use `-a filename` to attach a file

lynx and w3m

Text-based terminal web browsers

Example

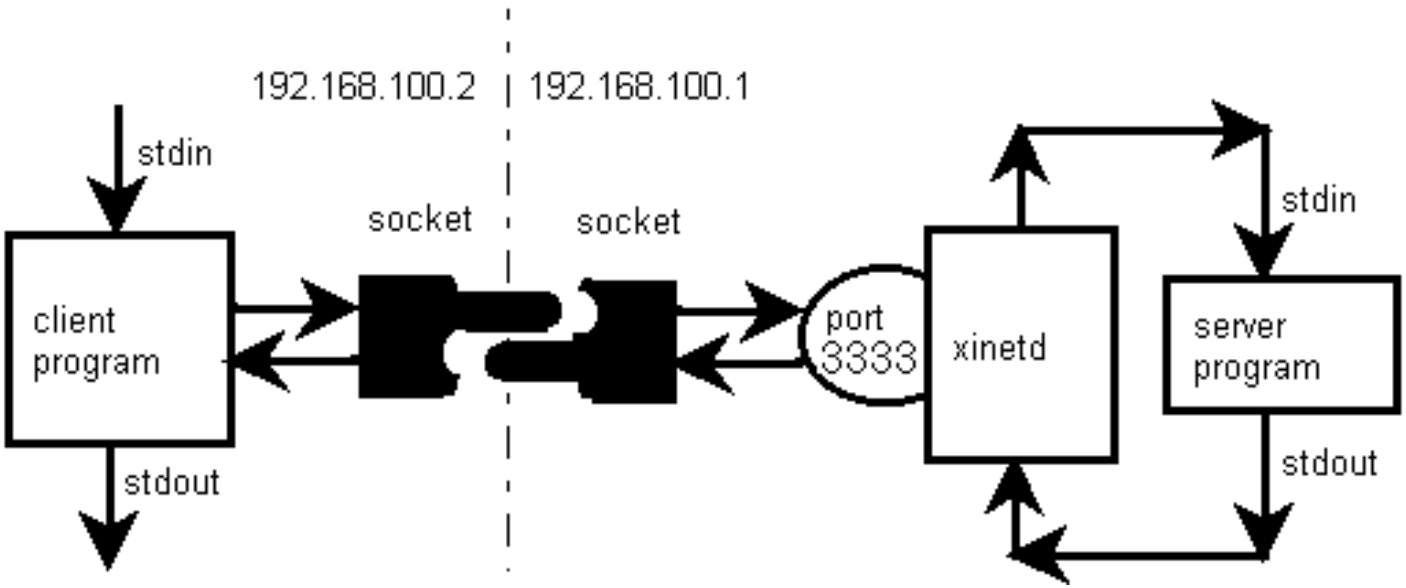
```
w3m [options] [URL or filename]
lynx [options] [path or URL]
```

Check the man page for more details

Sockets

A socket API allows application programs to control and use network sockets.
It makes talking to arbitrary machines around the world unbelievably easy.
A socket address is the combination of an **IP address and a port number** end of a telephone connection is the combination of a phone number and a particular extension.
Based on this address, internet sockets deliver incoming data packets to the appropriate application process or thread

xinetd(extended Internet services daemon)



python server example

```
# Echo server program
import socket

HOST = ''       # Symbolic name meaning the local host
PORT = 424242   # Arbitrary non-privileged port
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.bind((HOST, PORT))
s.listen(1)
conn, addr = s.accept()
print 'Connected by', addr
while 1:
    data = conn.recv(1024)
    if not data: break
    conn.send(data)
conn.close()
```

client example

```
# Echo client program
import socket

HOST = 'myhostname.domain'# The remote host
PORT = 424242       # The same port as used by the server
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect((HOST, PORT))
s.send('Hello, world')
data = s.recv(1024)
s.close()
print 'Received', repr(data)
```

watch

execute a program periodically, showing output fullscreen

Example

```
watch -n 5 ls
```

List the contents of a folder every 5 seconds

cron

`cron` is a program that enables unix users to execute commands or scripts automatically at a specified date/time

- `cron` is a daemon, which means it only needs to be started once and will lay dormant until it is required
- On most Linux distributions is automatically installed and entered into the start up scripts so you don't have to start it manually:
  - Check by tying `ps -e | grep cron`
  - Depending on your system, it may show up as `cron` or `crond`
- We can control the cron daemon in a few different ways...

cron and root

If you have a look in your `/etc` directory you will find sub directories called

- `cron.hourly`
- `cron.daily`
- `cron.weekly`
- `cron.monthly`

- If you place a script in any of these directories, it will be run either hourly, daily, weekly or monthly depending on the name of the directory.
- Note: If we did this with our backup script, we would need to replace $\sim$ with /home/hussam since the script would be run as `root`.

cron flexibility

If you want more flexibility in scheduling you can edit a `crontab` file

`crontab`

`crontab` files are cron's config files.

- The main config file is normally `/etc/crontab`
- You can create your own crontab files without root access!

Type `cat /etc/crontab` to have a look at the file:

main crontab

```
# /etc/crontab: system-wide crontab
# Unlike any other crontab you don't have to run the 'crontab'
# command to install the new version when you edit this file
# and files in /etc/cron.d. These files also have username fields,
# that none of the other crontabs do.

SHELL=/bin/sh
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin

# m h dom mon dow user command
17 * * * * root    cd / && run-parts --report /etc/cron.hourly
25 6 * * * root test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.daily )
47 6 * * 7 root test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.weekly )
52 6 1 * * root test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc cron.monthly )
#
```

crontab

Syntax:

```
a.    b.    c.    d.    e.    command to be executed
```

a. min (0-59)
b. hour (0-23)
c. day of month (1-31)
d. month (1-12)
e. day of week (0-6) (Sunday = 0)

Values can be * (all legal values), a range separated by a hyphen, a single value, a set of values separated by commas or a step value (i.e. */2 could be every two hours).

multiuser crontab

- To edit your crontab file type `crontab -e`
- To view your crontab file type `crontab -l`
- To delete your crontab file type `crontab -r`

A sample line:

```
30   18   *   *   *   ./home/backup.sh
```

This runs the backup script everyday at 6:30PM.

Unix tips: good place to place scripts

When you type a command name, bash searches for it in the directories specified in PATH

- Commands are searched in the order specified in PATH.

Example:

```
$ echo $PATH
/home/me/bin:/usr/local/sbin:/usr/local/bin
:/usr/sbin:/usr/bin:/sbin: /bin:
```

- Use the PATH variable to add directories to your search path.

Adding a directory

```
$ PATH=~/bin:"$PATH"
```

Next Time