

Network Administration and Security

- all notes are available from **blackboard** (<http://blackboard.glam.ac.uk>).

Defining terms

- these notes will mention UNIX and GNU/Linux
- when UNIX is mentioned it refers to all brands of *NIX and its clones
 - whereas GNU/Linux refers to the specific variety of UNIX clone

Seven cardinal virtues of system administration

- flexibility
- ingenuity
- attention to detail (security)
 - many large security problems display only tiny symptoms, which the inattentive system administrator will miss
- adherence to routine
 - example, do not forego security monitoring or backups
- persistence
 - monitoring security and system performance is an ongoing activity whose rewards are rarely immediate
- laziness! (see automating tasks)
- patience

General guidelines

- document how much time you spend caring for the system. Divide it into categories:
 - system maintenance
 - user support
 - routine activities
 - system enhancement
 - your own documentation!
- catastrophes *will* happen, all hardware has a finite lifetime
- encourage users to make their own backups

General guidelines

- peer pressure (printing, disk space, lack of backup etc)
- provide central repository for key files for backup
 - users can place files here if they are tight for time
- create tools for automating the backup process

Security guidelines

- passwords should be easy to remember but difficult to crack
- educate users how to choose a good password (produce a 1 page handout for example)
- do not build passwords with:
 - any part of your name or relatives name
 - significant numbers (telephone, date of birth, etc)
 - something of importance to you (pop star/group, TV character)
 - any name associated with your company or its products
- some defence contractors will fire you if a password is written down

Administration passwords should be changed

- whenever someone learns of it who should not!
- whenever a member of the administration team leaves
 - including all dial up lines!
- if you suspect the password file was read by an unauthorised person
 - in this case *all* passwords must be changed
- if you think someone was paying too much attention to your fingers
- regularly change passwords (period of time depends on the institution)

Automating tasks with scripts and tools

- a script is a program (or file) which is built from combining multiple commands:
 - example DOS bat file or UNIX shell script
- writing scripts is part of an administrator's job
- two main types:
 - designed to make system administration easier and more efficient, often by automating some process or job
 - provide useful tools for users which are not otherwise available to them

Advantages of automation

- greater reliability
 - tasks are performed in the same (correct) way every time
- guaranteed regularity
 - tasks can be performed according to whatever schedule seems appropriate
 - need not depend on your availability or even your presence!
- enhancing system efficiency
 - time-consuming or resource-intensive tasks can be performed during off hours
- popular scripting languages (expect, shell, perl) and associated tools (cron and find)
 - note expect, Bourne shell (bash) and perl are free

/etc/passwd

- is a text file containing user names and passwords
 - an example /etc/passwd/

```
root::0:0:root:/:/bin/bash
bilbo::100:100:Bilbo Baggins:/home/bilbo:/bin/bash
fred::101:100:Fred Bloggs:/home/fred:/bin/bash
```

- cut is a program which cuts files into columns

```
cut -d : -f 1 /etc/passwd
```

- when the above command line is run, it emits:

```
root
bilbo
fred
```

Simple shell scripts examples

- to find out which users use netscape we could run the following script

```
#!/bin/sh
for i in bob bill sue mary; do
  if [ -f /home/$i/.netscape ] ; then
    echo "$i is a user of netscape"
  else
    echo "$i has never used netscape"
  fi
done
```

- what problems exist with such a program?

Improved netscape analysis program

- what happens if users are added and deleted from /etc/passwd?
 - old program will fail, so:

```
#!/bin/sh
for i in `cut -d : -f 1 /etc/passwd` ; do
  if [ -f /home/$i/.netscape ] ; then
    echo "$i is a user of netscape"
  else
    echo "$i has never used netscape"
  fi
done
```

- is much better

Shell (bash) pros/cons

- pros
 - consider the cement which holds together the utilities of UNIX
 - very powerful, as it can execute any other program
 - use one programs output as input for another program
 - the main scripting language on a UNIX machine
 - looping constructs (for while)
 - conditional (if then else fi, case)
 - functions also allowed

- cons
 - cryptic
 - needs to be installed as an extra on non UNIX machines

Perl

- pros
 - perl is also freely available
 - perl is the de facto standard scripting language for WEB sites
 - runs on most operating systems
 - rich set of features, very powerful

- combines short development time of shell scripts with some of the best features of high level languages
 - many security features

- cons
 - too many features, many different ways of achieving same end!

Perl example

```
#!/usr/bin/perl

@users = `cut -d : -f 1 /etc/passwd`;
foreach $user (@users) {
  chop($user); # removes newline
  if (-f "/home/$user/.netscape") {
    print "$user uses netscape0 ;
  }
}
```

- perl is very good at working with arrays, strings etc

Expect

- *a software suite for automating interactive tools*

- expect allows you to drive interactive programs from a script
 - send commands to a program
 - expect output from the program
 - perform appropriate action based on the output received!

- very useful for automating tasks which require fairly predetermined keyboard interaction

- available on UNIX

Expect example applications

Python

- automating ftp retrieval
 - netscape allows you to download data
 - often times out!
- could use ftp
 - tedious, needs precise keyboard interaction
 - user may have to wait considerable time for each command
- fetch items at night, via expect!

- is free
- is between Java and Perl regarding tool set
- ease of use of scripting language but with tools associated with programming languages
- OO
- many libraries, modules, third party utilities
 - has an OO interface to the Tk GUI
- automatic garbage collection

Python example

Cron

- ```
#!/usr/bin/python
print 'hello world'
```

- when run will print the classic hello world

- ```
#!/usr/bin/python
from Tkinter import *
w = Button(text="hello world", command='exit')
w.pack()
w.mainloop()
```

- a program which will run other programs at user specified times.
- cron is controlled by the contents of the cron table (called crontab)
 - this is a file in /var/spool/crontab which may look like this:

- ```
here we run the cvs tests at 8:00 in the morning
#
00 08 * * * /usr/local/bin/runcvstests
#
here we remind the non Windows users to change
their password.
#
00 00 28 * * /usr/local/bin/mailfred
```

## Other time examples of cron

```

MIN HOUR DAY MONTH DAYOFWEEK COMMAND
at 6:10 a.m. every day
10 6 * * * date

at 11:00 a.m. on the 4th and on every
mon, tue, wed
0 11 4 * mon-wed date

4:00 a.m. on january 1st
0 4 1 jan * date

```

## Scripting languages

- many different languages: bash, perl, expect and others
- traditionally bash is used as the UNIX scripting language, however newer languages such as perl and python are becoming popular
  - bash, perl python run on almost all modern operating systems

## Example administration script in bash

```

#!/bin/sh
#
mailfred
(Should be run by cron once a month)
#
generateMessage
#
function generateMessage () {
 cat << EOFEOF | mail -s "**** a \
warning for Ludites ****" $1
}

Cron is sending you a message.

Remember to log into Windows and choose
another password.

EOFEOF
}

generateMessage fred

```

## Laboratory

- using a suitable editor (emacs) type in the following program:
- ```
#!/usr/bin/python
print 'hello world'
```
- save it as a filename hello.py

Executable scripts

- make this program executable via:
 - `chmod 755 hello.py`
 - the 755 is the permission of the file, set to execute and safe ownership
- now run your program by:
 - `./hello.py`
 - the ./ runs the program in your current directory and not another program called `hello.py` elsewhere on the GNU/Linux system
- you could easily write cgi programs using this method

Executable scripts

- Now type in the following program to an editor (hint use the emacs editor)
 - ```
#!/usr/bin/python

from Tkinter import *
w = Button(text="hello world", command='exit')
w.pack()
w.mainloop()
```
- make it run
- alter the message.
- if you have any time left over read the further course notes on Python and write a program which generates the 13 x table