File permissions: Protecting Your Files

- **ls –l *myfile.dat***:
  -rwxr-xr-x 1 userid gid 10 Feb 15 09:00 myfile.dat
- Octal/bit mapping: r w x r w x r w x
- 4 2 1 4 2 1 4 2 1
- Example: r-xr-xrwx ➔ 101 101 111
  4+1, 4+1, 4+2+1 = 557 (octal)
- permissions: user, group, others; flexible scheme
- to change used **chmod**: **chmod** *[-R] newperm file*
- symbolic form and (absolute) octal form:
  - **chmod og+w** *myfile* or
  - **chmod 777** *myfile*
- **chgrp**: change file group access: usually root
- **umask**: report/set file and dir creation permissions
  - Uses octal codes and xor operations!

Some Other useful Utilities

- most write to stdout read from stdin
- sort: **sort *myfile***, orders lexically
- **uniq**: removes duplicates in input
- **find**: very powerful...and mysterious (!)

  find . -name "*foo" -type f -exec rm {} \;
- **grep**: search files: grep "foobar" f1 f2 ... fN
- **wc**: count characters, lines, letters: **wc -l** *file* (count lines)
- **touch** *file* – update access time for file, or create 0 size file

Finding Out Who's About

- **who** list of users logged onto host
- **ps** list of processes; lots of options

  ps –al (a=all users, l=long listing)
- **finger** *X*: get info on user with login X
  - Hey, I saw that! 
- **top**: constantly updating list of process on system

Working on remote machines

- SSH – secure shell (logon to a remote machine)
  - Starts a session on specified host machine
    - ssh name@hostname or ssh –l name hostname
    - ssh –X/-Y setups up X11 forwarding (if permitted & X server running)
  - Can run in terminal window over slow connection
  - Can use "putty" (Google for this)
- sFTP – secure File Transfer Protocol (need secure ftp client)
  - Move files between machines
    - **sftp** nightmare.cs.uct.ac.za
    - Use **ls**, **pwd**, **cd** to navigate directories
    - Use **put** *file*/**get** *file* to upload/download file
    - **mget/mput** *filelist* gets/puts multiple files in one command
      - Use **prompt** to avoid having to confirm for each file…
    - Type **close** or **exit** to close connection
  - **anonymous** login (public archive - e-mail as password)

Help on Getting Started

- Read the manual pages!
- System Admin help: help@cs.uct.ac.za
- look at the web
  - Many UNIX resources (google!)
    http://linuxcommand.org/index.php
    http://www.tutorialspoint.com/unix/
    http://matt.might.net/articles/basic-unix/
    http://www.thegeekstuff.com/2010/11/50-linux-commands/
    http://freeengineer.org/learnUNIXin10minutes.html
  - CS course chatroom: vula.uct.ac.za
- don't print binary files!!
- **whoami**
  - when you're having a bad day ;-)

The Shell

- shell provides execution environment
- support notion of "jobs"
- job control:
  - **bg**, **fg**, **kill**, ^Z, &
  - *jobs* command – list jobs
- many kinds (sh, ksh, csh, zsh, bash...)
- each has benefits/and disadvantages (scripting, features)

Shell Customization

- ENV variables; eg PATH, HOME
- shell init files: .*shrc, .login, .profile
- changing your default editor:
  - setenv EDITOR emacs (for t/csh)
  - export EDITOR=emacs (for ksh)
- Default shell set by root

Shell Metacharacters

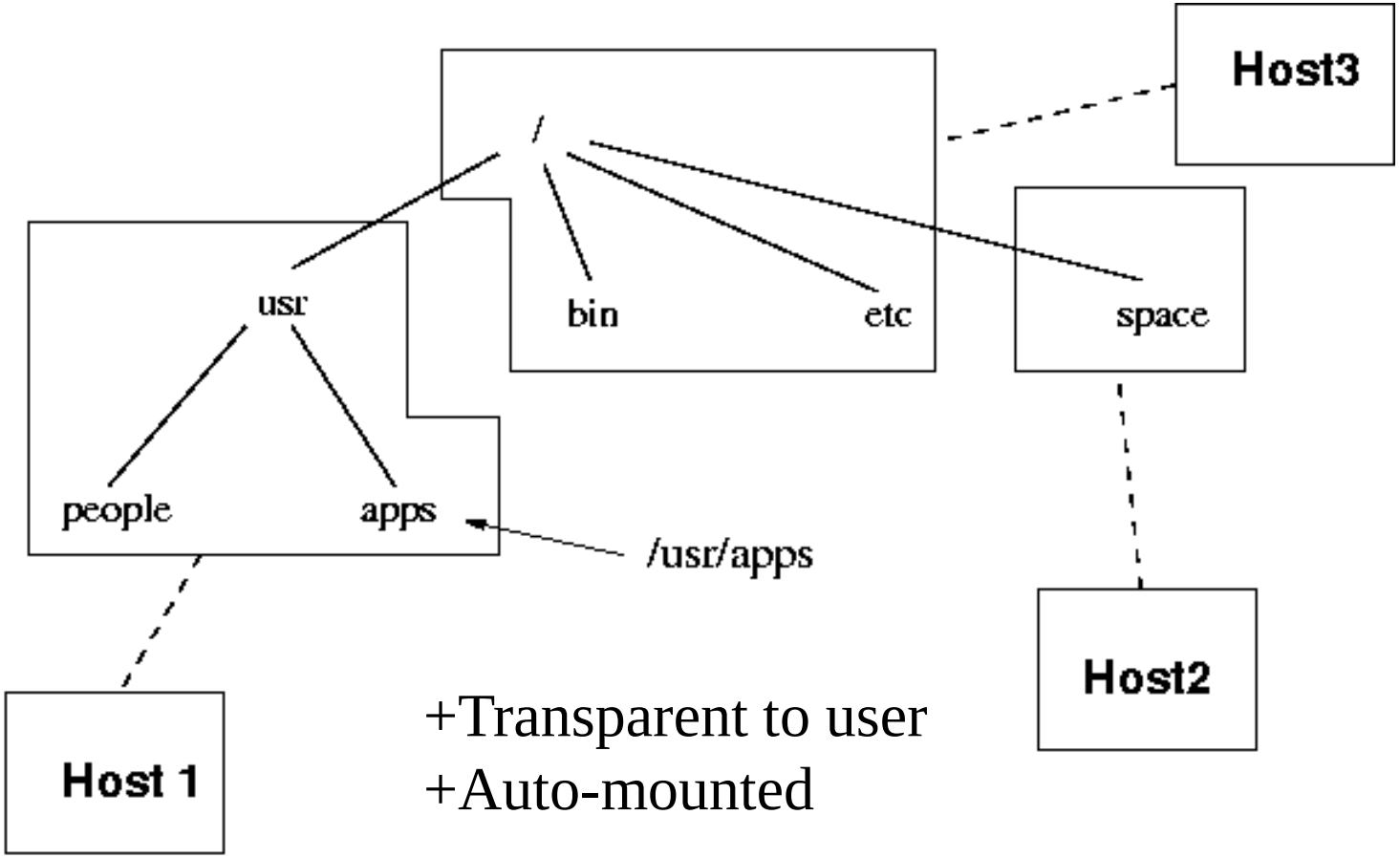
- Some characters reserved (like keywords in Java)
  - Include: *, ?, ; , {}, ()
- These have special meaning to shell
  - *, ? are used for filename expansion
  - {}, ; , () are used for processing commands
- do NOT use these in file names or on command line
- NB: space is used to separate input – files names should not have spaces
  - can use "" for names with spaces: ls –l "my file"
- To use special character, "escape" it with \
  - * or \; etc

The Unix File System

- files and directories arranged in a logical hierarchical *file system*
- Physically space may reside across a network
- Under Unix, a directory is just a special file
- files consist of fixed size *file blocks* (0.5-4KB)
- file info viewed with **ls -l**; inodes, links
- creating hard and soft links

  ln -s *file1 file2* ...file2 a "nickname" (soft link)
- disk usage: **du**
- file system space: **df** [*filesysname]*

Network File System (NFS)



+Transparent to user
+Auto-mounted

Device Files and I/O Redirection

- Unix device files ("special" files)
  - /dev/null - write data here to make it vanish!
  - /dev/audio – write data (sound) to play
  - /dev/console, /dev/ttyN – console/tty devices
  - only Some "devices" allow input/ouput

    echo "a b c" > /dev/ttyq0
- each process may have stdin, stdout
  - command < src > dest
    - **more** < *file*
    - **cat** > aaa

      The quick brown fox

      ^D
    - **ls -l** > filelist.dat

- append:>> e.g.

  cat file1 >> filelist.dat (append file1 at end)
- selective input: <<IDENTIFIER
  - sort <<end > data.out
  - Input:

    The
    Quick
    Brown
    end
  - data.out: Brown, The, Quick

I/O piping: |

- linking of commands: powerful technique

  cat myfile | sort | uniq | grep CS1 > output
- output of each command linked to input of next
- combine with redirection:

  gzip -dc archive.tar.gz | tar xvf - > /dev/null

  This unzips archive, and extracts files, writing all messages to the null device
- **stderr** always writes to terminal (unless redefined)
- under most shells, can "alias"' these combos:

  alias dir='ls -lF' (using ksh)

Processes

- a program in execution
- has its own address space - can crash in peace!
- can communicate via "pipes" with other processes
- process priorities (be nice!): **nice** and **renice**
- use ps to view processes (many options):
  - e.g. **ps -f**

```
UID     PID     PPID  C  STIME    TTY TIME  CMD
patrick 44222 44648 0  10:04:34  ttyq1    1:07
xemacs
```