Lecture 3

Users (the bane of the Systems Administrator)

- An account (user) is usually identified by a login name (user name).
- The system will internally identify each user by a user id (integer).
- A password or some other form of authentication is usually associated with an account.

- When selecting a username users should remember that in some instances the username is world viewable.

- This is particularly common in organisations that have single sign on i.e. centralised account management.

Users

- Users will usually also belong to a set of groups. Under UNIX, each user has a primary group and a set of supplementary group memberships.
- Files have associated with them an owner (uid) and a group (gid).
- Note that the system does not enforce that owners or groups for files actually exist.

- Systems have some way to map a user name to a user id. Under UNIX this is the job of the password file.
- Actually, the password information may be coming from sources other than a file, such as a NIS+ or a LDAP database.
- Other information sources depend on a module called PAM (Pluggable Authentication Module)
- We still conceptually talk about the password file.

The Password File

- The password file (/etc/passwd) contains an entry for each valid account. The fields are:
  - Username
  - Encrypted password placeholder
  - UID
  - GID
  - GECOS information
  - Home directory
  - Login shell
- tim:x:1000:1000:Tim Citizen,,,:/home/tim:/bin/bash
- Most operating systems support 32 bit user id's but your mileage varies largely due to file system implementation.

- The password file is world readable.
- To overcome problems with password cracking, the encrypted password plus additional account information is stored in a shadow password file which is readable only by the super user.
- This file is called /etc/shadow.
- Anything that reads this file needs to be run as root.
- The *login* program runs Set UID root to allow reading of such a file. In fact anything that reads this file needs to be run as root.

The shadow file

- Shadow password files (/etc/shadow) contains the following information;
  - user name.
  - encrypted password.
  - time last changed.
  - min time between change.
  - max time between changes.
  - warn this many days before password expires.
  - max days of inactivity after expiry before disabled (Solaris)
  - expire days since Jan 1 1970 that account is disabled. This specified when the account may no longer be used. There is also an additional field which is reserved.
- username:passwd:last:may:must:warn:expire:disable:rsvd

- UNIX uses a simple one-way encryption algorithm (hash) to encrypt the passwords. Implemented by *crypt()*.
- When a plain text password is entered, it is encrypted and the results compared with the value in the shadow password file.
- There is no way to find the plain text password from the encrypted password other than by search.

- As machines have become faster, it has become possible to perform the encryption step fast enough that people can try a reasonable number of plain text passwords.
- This meant that the fact that the plain text passwords were visible became a liability.
- Hence the invention of the shadow password file.

- The public domain utility “crack” tries to guess passwords given the encrypted versions.
- It uses dictionaries and heuristics to guess viable passwords.
- Exhaustive search is still not feasible unless you have a lot of hardware.

- Some systems use a different hash which is slower to compute than the UNIX *crypt* function.
- This slows down search speed when cracking passwords.
- Examples are MD5 or SHA1 cryptographic checksums. (Linux and BSD can do this).
- We are beginning to see this technology apply in the file system space as well.

- The password comprises of parts: 1buJ6v3Ch$BwlfdhdheywydwwD.hdhdjsjw
- The first is the cryto system used:
  - 1 means MD5. If there is no leading $ it uses the standard DES.
- The password field can be 13 to 24 characters from a 64 character alphabet i.e. a-zA-Z0-9/ and .

- In Linux, if the password contains leading !! or is simple :!!: then this means the account is not active
- You can restrict account by using */sbin/nologin* as the shell but again it varies from platform to platform.
- The remaining parameters are set by the *passwd* () program and depending on the OS can be changed from the default by editing a file like /etc/login.defs

Login.defs

- In Linux, to change password policy through command line in Linux, we just have to edit */etc/login.defs* file. Only root user can edit this file.

```
MAIL_DIR        /var/spool/mail

PASS_MAX_DAYS   99999
PASS_MIN_DAYS   0
PASS_MIN_LEN    5
PASS_WARN_AGE   7

UID_MIN         500
UID_MAX         60000

GID_MIN         500
GID_MAX         60000

CREATE_HOME     yes
```

- When *useradd* command is used a user is created. The *useradd* binary now proceeds with the process of user creation and goes to the */etc/login.defs* file to get following values from the file :

1. MAIL_DIR: Directory where the user's mail will be stored.
2. PASS_MAX_DAYS: Maximum number of days for the validity of a password.
3. PASS_MIN_DAYS: Minimum number of days gap before a password can be changed again.
4. PASS_MIN_LEN: Minimum required length of a password.

5. PASS_WARN_AGE: Warning for password expiry to be given before the stipulated number of days.
6. UID_MIN: Minimum value for automatic user id selection.
7. UID_MAX: Maximum value for automatic user id selection.
8. GID_MIN: Minimum value for automatic group id selection.
9. GID_MAX: Maximum value for automatic group id selection.
10. CREATE_HOME: Whether *useradd* should create home directories for users .

Directory service

- Some systems allow for encrypted passwords and other account information to be accessed via directory services such as:
  - LDAP
  - NIS
  - NIS+
  - Active Directory

PAM

- Pluggable Authentication Modules (PAM) enables a system of shared libraries for authentication
- To authenticate a user, an application hands the user to PAM to determine whether the user can successfully identify itself.
- Because there are many kinds of authentication scenarios, PAM employs a number of dynamically loadable authentication modules. Each module performs a specific task; for example, the pam_unix.so module can check a user's password.

- LDAP and Kerberos together make for a great combination. Kerberos is used to manage credentials securely (authentication) while LDAP is used for holding authoritative information about the accounts, such as what they're allowed to access (authorization), the user's full name and uid. You can also add in helpful things such as an external email address or a room number in a structured way.
- https://wiki.debian.org/LDAP/Kerberos

Password Selection…

- Left to themselves users will:
  - Never change their password.
  - Choose really stupid ones that are easy to guess.
  - Cycle their passwords.
  - Write them down.
  - Attempt to circumvent you stopping them doing 1,2,3 and 4.
  - Complain

- The administrator must work within the confines of human nature.
  - If you make passwords too hard to remember, users will write them down.
  - If you make them change them too often, they will cycle them.
- Base your policies on the situation and your security requirements.

User Organization

- The next important piece of information in the password file is the group information.
- Have some mechanism for grouping your users.
- Do this in such a way it is easy to determine which user group any particular account belongs to.
- This allows you to easily impose differing quotas/limits etc. on your user groups.

Groups

- Groups (as the name implies) are a way of grouping your users together.
- Most systems allow file access to be controlled by groups, thereby giving a mechanism for file sharing.
- Avoid complicated group structures if possible.
- Administration becomes cumbersome and costly in SA time.

- Users will also have primary groups (GID).
- They will also belong to other secondary groups which are typically enumerated in */etc/group*.
- name:password:GID:members
  - Group name may be a maximum of 8 characters
  - Group password can be a place holder
  - Each group has a unique identifier
  - Members field indicates the list of user accounts that are members of the group

- The general form of /etc/group is:

  wheel:x:10:trent,bob,sam

  cs-staff:*:100:tim,risque
- From the file you can see group name, group id and membership, what about the x and *.
- Groups can have passwords (2nd field) which allow non-members to change to it using the *newgrp* command.
- If the password is an x it means consult */etc/gshadow*.
- A * in the password field means there is no password.

View and temporarily change your primary group

- To display the groups that you are a member of
  - The *groups* command
- To further display UID, primary GID, and all GIDs for the groups that you belong to
  - The *id* command
- To change your primary group temporarily
  - The *newgrp* command

Home directories

- Each user will normally have their own home directory.
- Choose a home directory naming scheme which reflects your user grouping structures.
- For example at the University we have groups and home areas tied to each group.
- This allows you to easily identify which group of users will be affected by file system outages an
po

User group	Home directory
Staff	/home/staff
Guests	/home/guests

Adding users: the basic steps

- Having the new user sign your policy agreement
- Edit the *etc/passwd* and *etc/shadow* files to define the user's account
- Add the user to the */etc/group* file
- Set an initial password
- Create the user's home directory
- Copy startup files to the user's home directory
- Set up mail alias (it is convenient for each user to receive mail on only one machine)
- Update LDAP database (directory services)

Creating user accounts

- If you have to add a user by hand, use *vipw* to edit the passwd and shadow files.
- Set the password for the new user with the *passwd* command.
- You can create the new user's home directory with a simple *mkdir* command
- You will need to set the ownership and permission by using *chown* command
- You should run the *pwck* command to check the format of /etc/passwd and ensure that no structural errors have occurred

Startup files

- Startup files traditional begin with a dot and end with the letters rc.
- Provide default startup files for each shell that is popular on your system, e.g.,
  - .profile for sh
  - .bashrc and .bash_profile for bash
- Startup files typically set a default environment including
  - Search path
  - Terminal type and environemnt
  - Command aliases
  - umask value

- Startup files are traditionally kept in */etc/skel* or */etc*
- If you customise your vendor's startup files, */use/local/etc/skel* is a reasonable place to put the modified copies.
- */etc* may contain system-wide startup files that are processed before the user's own startup files.
- Users can override your settings in their own startup files

Final steps

- To verify that a new account has been properly configured, first log out, then login in as th new user and execute the following commands:
  - pwd (to verify the home directory)
  - ls –la (to check owner/group of startup files)
- Notify new users of their login names and initial passwords
- Remind new users to change their passwords immediately

The *useradd* command

- Most systems supply you with an easy to use mechanism for creating accounts.
- Ubuntu provides two ways to add users: *adduser* and *useradd . adduser* is a perl wrapper for *useradd.*

  $sudo useradd tim

  $ sudo useradd –c "Tim Citizen" –d /home/tim –g faculty –G famous –m –s /bin/bash tim
- *useradd* disables the account by putting an x in the password field, you must assign a real password to make the account usable.
- *adduser* has a twin *addgroup* and cousins *deluser* and *delgroup*.

Option	Description
-c “description”	Adds a description for the user to the GECOS field
-d homedir	Specifies the user’s home directory
-e expirydate	Specifies the date to disable the account
-f days	Specifies the number of days that an inactive account will be disabled
-g group	Specifies the primary group for the user
-G group1, group2, etc.	Specifies all other group memberships for the user
-i	When used with the –u option, it allows a UID that is being aged to be used

Option	Description
-k directory	Specifies the skeleton directory used when copying files to a new home directory
-m	Specifies that a home directory should be created for the user account
-o	When used with the –u option, it allows a UID to be duplicated on the system
-s shell	Specifies the absolute pathname to the shell used for the user account
-u UID	Specifies the UID of the user account

Adding users in bulk (Linux)

- Linux's *newusers* creates multiple accounts at one time from the contents of a text file.
- The input file should have lines just like the */etc/passwd* file, except that the password filed contains the initial password in clear text.
- You are probably better off writing your own wrapper for *useradd* in Perl or Python than in trying to use *newusers*

Modifying user accounts

- The *usermod* command can be used to safely modify most information regarding user accounts

Option	Description
-c “description”	Specifies a new description for the user to the GECOS field
-d homedir	Specifies the user’s new home directory
-e expirydate	Specifies the date to disable the account
-f days	Specifies the number of days that an inactive account will be disabled
-g group	Specifies a new primary group for the user
-G group1, group2, etc.	Specifies all other group memberships for the user

Option	Description
-l name	Specifies a new login name
-s shell	Specifies the absolute pathname to the new shell used for the user account
-u UID	Specifies a new UID of the user account

Disabling logins

- Put a star or some other character in front of the user's encrypted password in */etc/shadow*
- On Linux, *usermod –L user* to lock password, *usermod –U user* to unlock passwords
- Does not notify the user of the account suspension
- Replace the user’s shell with a program that prints an explanatory message and supplies instructions for rectifying the situation.
- Check if a user’s login shell is listed in */etc/shells*

Removing users

- If you remove a user by hand
  - Remove the user from any local user databased or phone lists
  - Remove the user from the aliases file or add a forward address
  - Remove the user *crontab* file and any pending at jobs or print jobs
  - Kill any of the user's processes that are still running
  - Remove the user from the passwd, shadow, group, and gshadow files
  - Remove the user's home directory
  - Remove the user's mail spool

The *userdel* command

- The *userdel* command removes entries from both */etc/passwd* and */etc/shadow* corresponding to the user account
- -r option removes the home directory
- Any files that were previously owned by the user become owned by the number that represents the UID of the deleted user
- Any future user account that is given the same UID then becomes the owner of those files

Managing groups

- Edit */etc/group* file using a text editor
- You should run *grpck* command to check the format of */etc/group* and ensure that no structural errors have occurred
- There are commands similar to user management, e.g., *groupadd*, *groupmod*, and *groupdel*

Managing accounts

- Again depending on the OS there are lots of little tools to manage users and groups - some examples include:
  - *adduser/ useradd*
  - *userdel*
  - *groupadd*, *groupmod*, *groupdel*
  - *chage* (changes */etc/shadow* info)

Traditional UNIX access control

- Objects have owners. Owners have broad control over their objects.
- You own new objects that you create.
- The special account called “root” can act as the owner of any object.
- Only root can perform certain sensitive administrative operations.

Filesystem access control

- Every file has both an owner and a group (aka group owner)
- The owner can set the permissions of the file for
  - The owner
  - The group
  - Everyone else
- The permission is defined as read, write, and execute
- Both the kernel and the filesystem track owners and groups as numbers rather than as text names

The root account

- The root account is the superuser account
- The root account has a UID of 0
- Example of root operations
  - Changing the root directory of a process with *chroot*
  - Creating device files
  - Setting the system clock
  - Raising resource usage limits and process priorities
  - Setting the system's hostname
  - Configuring network interfaces
  - Opening privileged network ports (below 1,024)
  - Shutting down the system

Role based access control

- A layer of indirection is added to access control calculations.
- Roles are similar to groups but can be used outside of the context of the filesystem.
- Roles can have a hierarchical relationship to one another.
- Solaris can manipulate roles using *roleadd*, *rolemod*, and *roledel*
- Is the role the same as group?

The root account

- You need to keep the root account password safe.
  - Change the root password regularly.
  - Avoid using the root password over networks.
  - Keep it secure (written down in and in a safe).
  - Avoid it like the plague.

su: substitute user identity

- If invoked without arguments, *su* prompts for the root password and then starts up a root shell.
- Root privilege remain in effect until you terminate the shell.
- The *su* command can also substitute other identities other than root: *su* –username
- Type the full pathname suchas */bin/su* rather than *su*
- *su* has been largely superseded by *sudo*

sudo: limited su

- *sudo* is a package that allows you to delegate root powers or a subset to conventional users.
- Users authenticate to the *sudo* program and then can do what they wish providing it fits in certain parameters.
- *sudo* runs setuid root but consults a file called */etc/sudoers*.
- *sudo* prompts user's own password
- Additional *sudo* commands can be executed until 5 minutes have elapsed with no further *sudo* activity.
- *sudo* keeps a log of *sudo* activities.

- The file has directives such as this:
  - The users to whom the line applies
  - The hosts on which the line should be heeded
  - The commands that the specified users can run
  - The users as whom the commands can be executed

ALL ALL=/dialup,/hangup

This directive allows all users to execute the commands *dialup* and *hangup*.

jim ALL=/dialup,/hangup

In this case the user Jim can run the commands.

%admin ALL=(ALL) ALL

Members of the admin group may gain root privileges

- If you would like the user *tim* to do things as root you could go;

  tim ALL=(root) ALL

- You can do many things with *sudo* e.g. define groups and other properties.

- To modify */etc/sudoers*, you use the *visudo* command

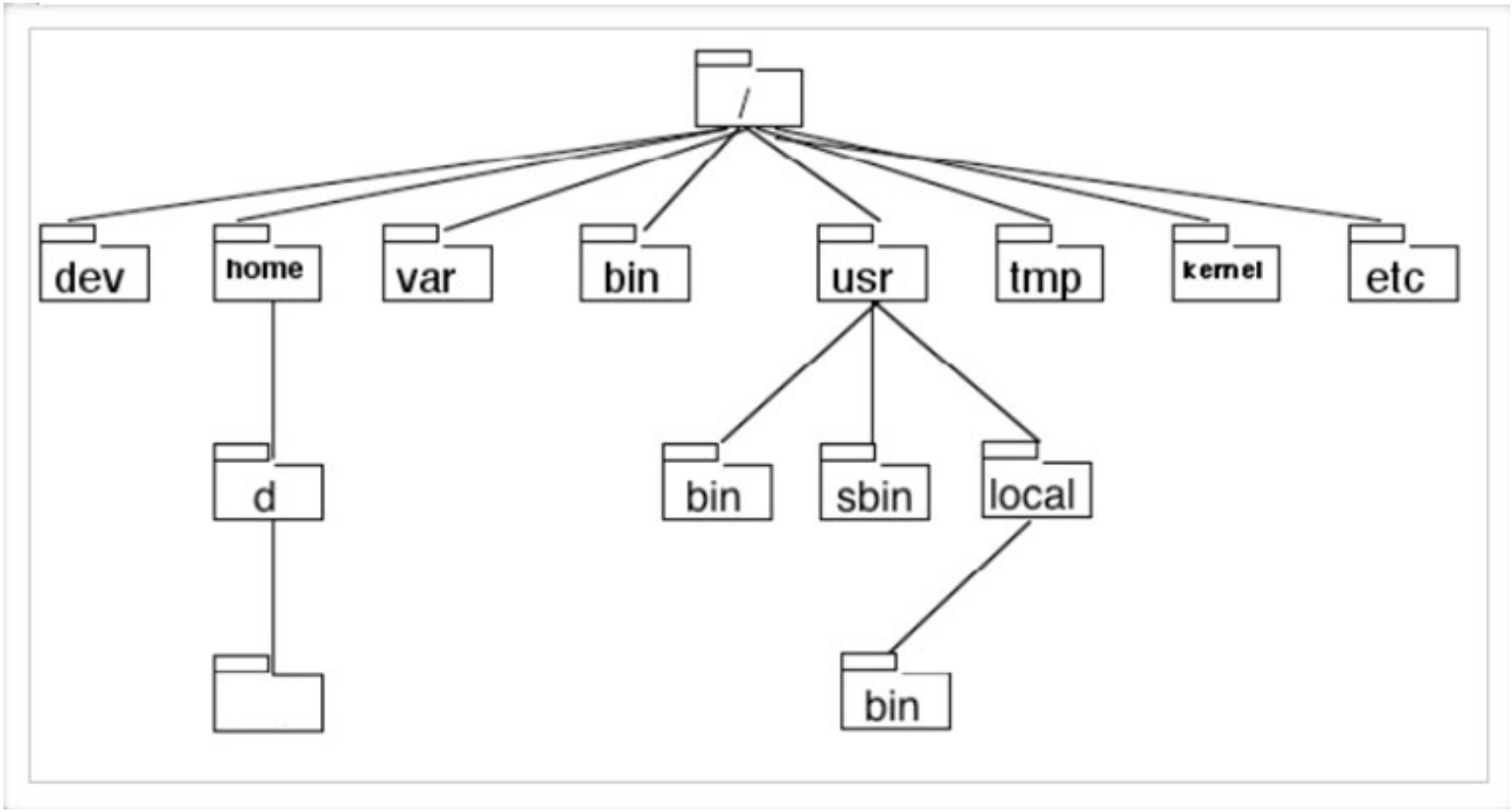
Advantages of using *sudo*

- Accountability is improved
- Operators can do chores without unlimited root privileges
- The real root password can be known to only few people
- It's faster to use *sudo*
- Privileges can be revoked without changing the root password
- There is less chance of a root shell being left unattended

The Filesystem

- The basic purpose of the file system is to represent and organise the systems storage resources.
- File systems on various partitions (logical or physical) are combined to form the real file system.
- The top level directory is often called root ‘/’.
- The file system structure result in it becoming an inverted tree structure.

A typical filesystem structure



- Objects in the file system can be identified by using their absolute or relative pathing.
  - Absolute pathing is the entire path through the file system to get to the actual object.
  - Relative pathing is the path ‘relative’ to the current working directory.

The Filesystem Hierarchy Standard (FHS)

- www.pathname.com/fhs/

Directory	Description
/bin	Binary commands for use by all users
/dev	Device files
/etc	System-specific configuration files
/home	Default location for user home directories
/lib	Shared program libraries
/mnt	Empty directory used for accessing disks such as floppy disks and CD-ROMs
/opt	Additional software programs

Directory	Description
/proc	Process and kernel information
/root	Root user’s home directory
/sbin	System binary commands (used for administration)
/tmp	Temporary files created by programs
/usr	System commands and utilities (/usr/bin, /usr/games/, /usr/include/, /usr/lib, /usr/local, /usr/sbin/, /usr/share/, /usr/src/, /usr/X11R6
/var	Log files and spools

Filesystem type

- Many filesystems are available for use in UNIX
- You need not use only one type of filesystem on the system
- Files and directories appear the same throughout the directory tree regardless of whether the UNIX system uses 1 filesystem or 20 different filesystems.
- Some common UNIX filesystems are:

Filesystem	Description
BFS	Boot Filesystem – a small filesystem used to hold files necessary for system startup
CDFS	Compact Disk Filesystem – used to view all tracks and data on a CD-ROM as normal files
DOSFS	DOS FAT Filesystem – used to access Windows floppy disk
HFS	High Performance Filesystem – used in HP-UX systems, based on S5
NFS	Network Filesystem – used to access filesystems on UNIX computers across a network
NUCFS	Netware-UNIX Client Filesystem – used to access Novell Netware filesytems from across a network

Filesystem	Description
S5	System V Filesystem – supported on all UNIX flavors
SFS	The Secure Filesystem – a variant of UFS that allows for secure access to files on the filesystem
UFS	The UNIX Filesystem – an improved version of the S5 filesystem and the default for Solaris
VxFS	The Veritas Filesystem – a journaling filesystem that offers large file support and extend-based allocation, and supports ACLs. It is also known as the Journaling Filesystem (JFS) and is the default filesystem on UnixWare and HP-UX systems

Check the filesystem type

- *df* Command – Find Filesystem Type.
- *fsck* – Print Linux Filesystem Type.
- *lsblk* – Shows Linux Filesystem Type.
- *mount* – Show Filesystem Type in Linux.
- *blkid* – Find Filesystem Type.
- *file* – Identifies Filesystem Type.
- *fstab* – Shows Linux Filesystem Type.

Mounting and unmounting

- A filesystem can be
  - A disk partition
  - A disk-based logical volume
  - A network file server
  - A kernel component
  - A memory-based disk emulator
  - Etc.

Filesytem mounting and unmounting

- File systems are attached to the tree with the *mount* command. *mount* maps a directory within the existing file tree, called the mount point, to the root of the newly attached filesystem. For example,

  mount /dev/sda4 /users

  installs the filesystem store on the disk partition /dev/sda4 under the path /users

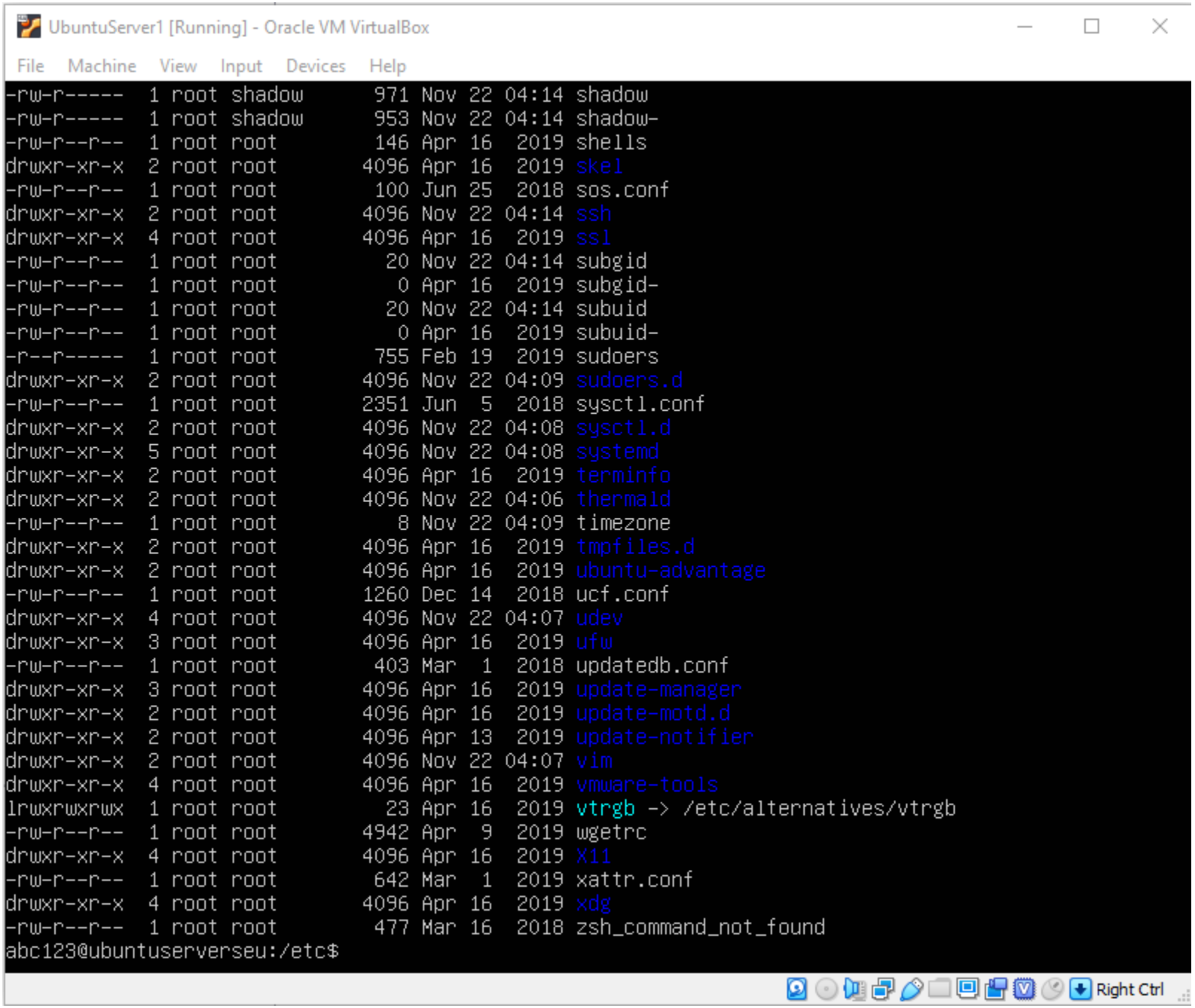
- To unmount the file system we use the *unmount* command.
- The kernel keeps track of what process file descriptors as associated with
- As a consequence you cannot unmount a file system that has a object opened by a process.
- Except of course if you use the -f (force unmount option).

- You can find out who is using what file systems by using either;
  - *lsof* is a open source program which peeks into kernel structures for all processes. It can report what files any process has open, file descriptors and even network sockets.
  - *fuser* returns the PID's of processes that happen to have open file descriptors. It initially appeared in FreeBSD but now is mainstream.

File types

- Regular files (-)
- Directories (d)
- Character device files (c)
- Block device files (b)
- Local domain sockets (s)
- Named pipes (FIFOs) (p)
- Symbolic links (l)

- You can determine the file type with *ls –l*. The first character of the output encodes the type.


UbuntuServer1 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
-rw-r----- 1 root shadow 971 Nov 22 04:14 shadow
-rw-r----- 1 root shadow 953 Nov 22 04:14 shadow-
-rw-r--r-- 1 root root 146 Apr 16 2019 shells
drwxr-xr-x 2 root root 4096 Apr 16 2019 skel
-rw-r--r-- 1 root root 100 Jun 25 2018 sos.conf
drwxr-xr-x 2 root root 4096 Nov 22 04:14 ssh
drwxr-xr-x 4 root root 4096 Apr 16 2019 ssl
-rw-r--r-- 1 root root 20 Nov 22 04:14 subgid
-rw-r--r-- 1 root root 0 Apr 16 2019 subgid-
-rw-r--r-- 1 root root 20 Nov 22 04:14 subuid
-rw-r--r-- 1 root root 0 Apr 16 2019 subuid-
-r--r----- 1 root root 755 Feb 19 2019 sudoers
drwxr-xr-x 2 root root 4096 Nov 22 04:09 sudoers.d
-rw-r--r-- 1 root root 2351 Jun 5 2018 sysctl.conf
drwxr-xr-x 2 root root 4096 Nov 22 04:08 sysctl.d
drwxr-xr-x 5 root root 4096 Nov 22 04:08 systemd
drwxr-xr-x 2 root root 4096 Apr 16 2019 terminfo
drwxr-xr-x 2 root root 4096 Nov 22 04:06 thermald
-rw-r--r-- 1 root root 8 Nov 22 04:09 timezone
drwxr-xr-x 2 root root 4096 Apr 16 2019 tmpfiles.d
drwxr-xr-x 2 root root 4096 Apr 16 2019 ubuntu-advantage
-rw-r--r-- 1 root root 1260 Dec 14 2018 ucf.conf
drwxr-xr-x 4 root root 4096 Nov 22 04:07 udev
drwxr-xr-x 3 root root 4096 Apr 16 2019 ufw
-rw-r--r-- 1 root root 403 Mar 1 2018 updatedb.conf
drwxr-xr-x 3 root root 4096 Apr 16 2019 update-manager
drwxr-xr-x 2 root root 4096 Apr 16 2019 update-motd.d
drwxr-xr-x 2 root root 4096 Apr 13 2019 update-notifier
drwxr-xr-x 2 root root 4096 Nov 22 04:07 vim
drwxr-xr-x 4 root root 4096 Apr 16 2019 vmware-tools
lrwxrwxrwx 1 root root 23 Apr 16 2019 vtrgb -> /etc/alternatives/vtrgb
-rw-r--r-- 1 root root 4942 Apr 9 2019 wgetrc
drwxr-xr-x 4 root root 4096 Apr 16 2019 X11
-rw-r--r-- 1 root root 642 Mar 1 2019 xattr.conf
drwxr-xr-x 4 root root 4096 Apr 16 2019 xdg
-rw-r--r-- 1 root root 477 Mar 16 2018 zsh_command_not_found
abc123@ubuntuserverseu:/etc$
Right Ctrl

- Programs communicate with the system's hardware using device files.
- Character device files allow their associate drivers to do their own input and output buffering
- Block device files are used by drivers that handle I/O in large chunks and want the kernel to perform buffering for them.
- Most systems today implement some form of automatic device file management. On Linux distributions, */dev* is a standard directory for device files.

- Sockets are connections between processes that allow processes to communicate hygienically.
- Local domain sockets are accessible only from the local host and are referred to through a filesystem object.
- Created with the socket system call and removed with the *rm* command.
- Named pipes also allow communication between two processes running on the same host.
- Created with *mknod* and removed with *rm*.

- A symbolic link points to a file by name.
- It redirects its attention to the pathname stored as the contents of the link.
- A hard link is a direct reference while a symbolic link is a reference by name.
- Links can be created with the *ln* command

Filesystem structure

- A filesystem has
  - The superblock which defines the information about the filesystem in general
  - The inode table which consists of several inodes, each describing one file or directory and containing a unique inode number for identification. The inode also stores information such as the file size, data block locations, last date modified, permission and ownership.
  - The data that makes up the contents of the file as well as the filename are stored in data blocks that are referenced by the inode.

- Hard linked files are direct copies of one another as they share the same inode and inode number.
- If one file is modified, all hard linked files will be updated as well.
- Deleting a hard linked file does not delete all the other hard linked files.
- The command *ln file1 file2* will make a hard link to file1 called file2.

- Symbolic links do not share the same inode and inode number with their target file.
- The data block in a symbolically linked file contain only the pathname to the target file.
- When you edit a symbolically linked file, you are actually editing the target file.
- If the target file is deleted, then the symbolically linked file servers no function
- The command *ln -s file1 file2* will make a symbolic link to file1 called file2.

List the inode number of the file

- To display the inode number of a file, you can run
  - *ls –i /etc/passwd*
  - *stat /etc/passwd*

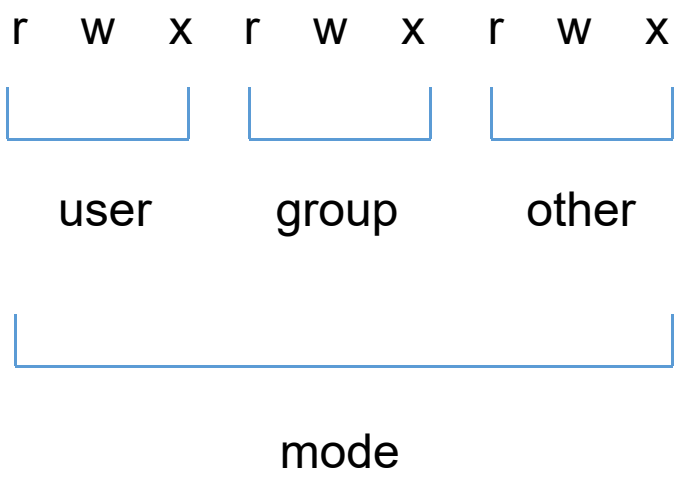
File ownership

- When a user creates a file, that user's name and primary group becomes the owner and group owner of the file.
- The owner and the root can change the ownership of the file (and the permissions on the file)
  - *chown user1 file1*
  - *chgrp group1 file1*
- Normally you should change both
  - *chown user1:group1 file1*

File permissions

- Permissions on a file is stored in the inode of the file
- It is called the mode and divided into three sections
  - User (owner) permissions
  - Group permissions
  - Other permissions
- The regular permissions are
  - Read
  - Write
  - Execute

Interpreting the mode



Changing permissions

- Use *chmod criteria filename* to change the permissions
  - *chmod u+w,g+r-w,o+r-x file1*
  - *chmod u=rw,g=r,o=r file2*
  - *chmod a+x file3*
  - *chmod +x file4*
- Ensure no space between any criteria
- Criteria can be numeric
  - read = 4
  - write = 2
  - execute = 1
  - rwx = 4+2+1 =7

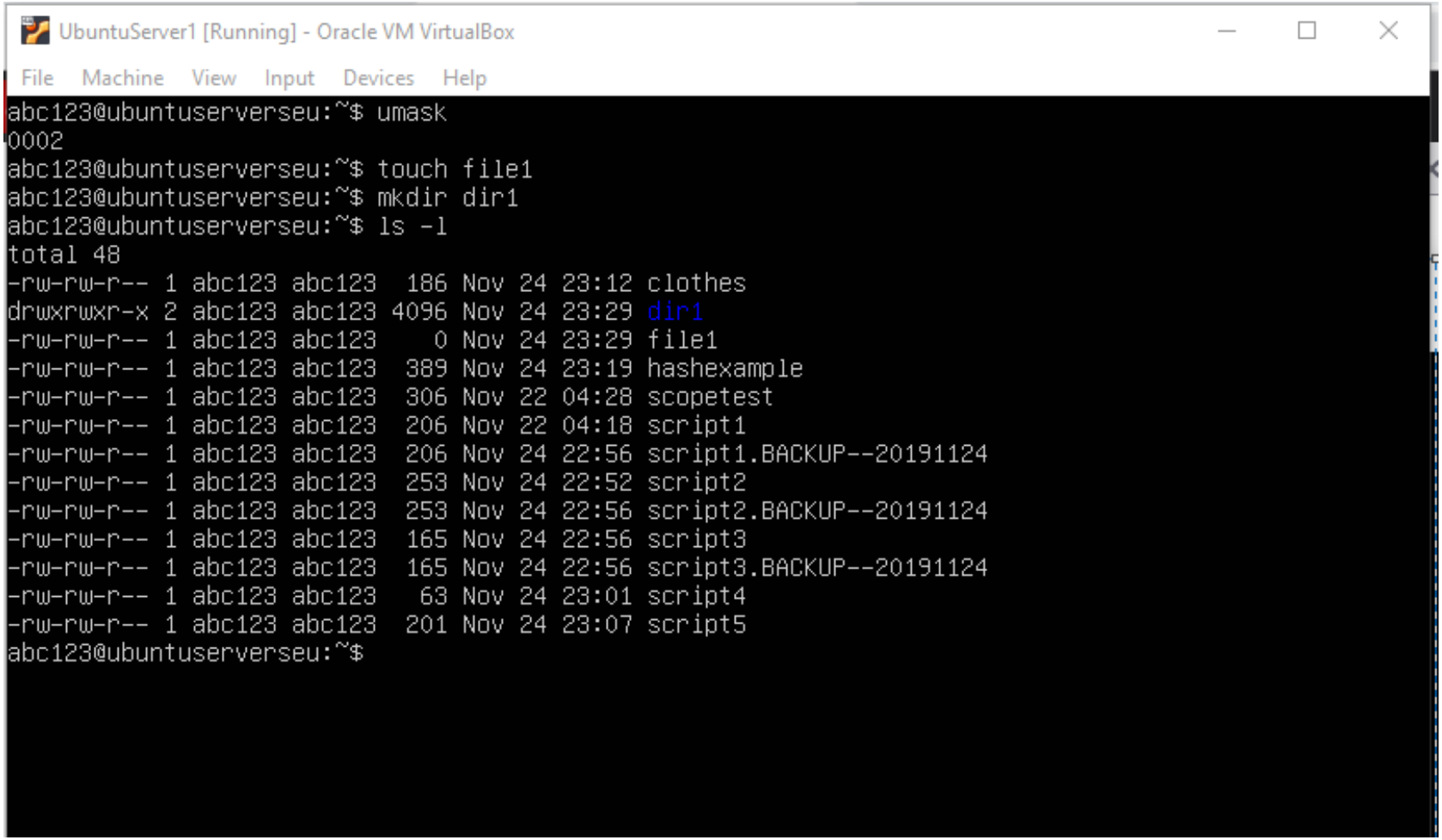
Directory permissions

- The read bit allows the affected user to list the files within the directory
- The write bit allows the affected user to create, rename, or delete files within the directory, and modify the directory's attributes
- The execute bit allows the affected user to enter the directory, and access files and directories inside
- The sticky bit states that files and directories within that directory may only be deleted or renamed by their owner (or root) (see slide 88)

- chmod 540 file1
- chmod 644 file2
- chmod 777 file3

Default permissions

- By default, new files are given rw-rw-rw-
- By default, new directory are given rwxrwxrwx
- A special variable umask takes away permission on new files and directories immediately after they are created
- The most common umask value is 022. what does it mean?
- When you log in and obtained a new shell, the default umask is loaded
- When you exit, your umask is destroyed


UbuntuServer1 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
abc123@ubuntuserverseu:~$ umask
0002
abc123@ubuntuserverseu:~$ touch file1
abc123@ubuntuserverseu:~$ mkdir dir1
abc123@ubuntuserverseu:~$ ls -l
total 48
-rw-rw-r-- 1 abc123 abc123 186 Nov 24 23:12 clothes
drwxrwxr-x 2 abc123 abc123 4096 Nov 24 23:29 dir1
-rw-rw-r-- 1 abc123 abc123   0 Nov 24 23:29 file1
-rw-rw-r-- 1 abc123 abc123 389 Nov 24 23:19 hashexample
-rw-rw-r-- 1 abc123 abc123 306 Nov 22 04:28 scopetest
-rw-rw-r-- 1 abc123 abc123 206 Nov 22 04:18 script1
-rw-rw-r-- 1 abc123 abc123 206 Nov 24 22:56 script1.BACKUP--20191124
-rw-rw-r-- 1 abc123 abc123 253 Nov 24 22:52 script2
-rw-rw-r-- 1 abc123 abc123 253 Nov 24 22:56 script2.BACKUP--20191124
-rw-rw-r-- 1 abc123 abc123 165 Nov 24 22:56 script3
-rw-rw-r-- 1 abc123 abc123 165 Nov 24 22:56 script3.BACKUP--20191124
-rw-rw-r-- 1 abc123 abc123   63 Nov 24 23:01 script4
-rw-rw-r-- 1 abc123 abc123 201 Nov 24 23:07 script5
abc123@ubuntuserverseu:~$

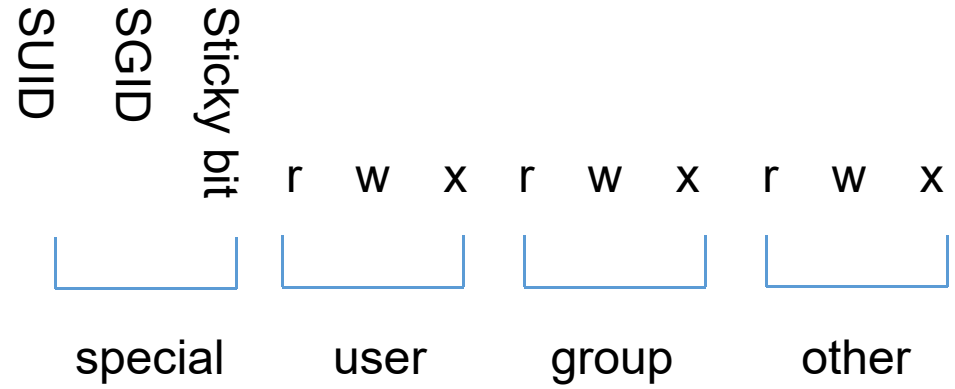
Changing the ACL

- In addition to user, group, and other, you may change the Access Control List by using the *setacl* command (*setfacl* in Linux)
  - setfacl –m u:lisa:r file
  - setfacl –x g:staff file
  - getfacl file1 | setfacl --set-file=- file2

Special permissions

- Set user ID (SUID)
  - If set on a file that is executable, the person who executed the file temporarily becomes the owner of the file while it is executing.
- Set group ID (SGID)
  - If set on a file that is executable, the person who executed the file temporarily becomes a member of the group that is attached to the file during execution.
- Sticky bit
  - If set on a directory in addition to the write permissions, the users may add files to the directory but only delete those files that they have added and not others.

Setting special permissions



- chmod 6755 file1
- chmod 1777 dir1
- chmod 1770 dir1

- Mask the execution permissions from rwxrwxrwx to
  - rwsrwsrwt
  - rwSrwSrwT

Questions?