PRONET

VNU-ITP

UNDP

Files and Directories

Objectives

PRONET

VNU-ITP

UNDP

- Describe and explain the Linux file system directory structure
- Explain file system concepts
- Identify and explain inodes
- Utilize `chown`, `chgrp`, and other file related commands
- Set user and group ID permissions
- Identify and explain sticky bits
- Identify and explain links

Standard Directories

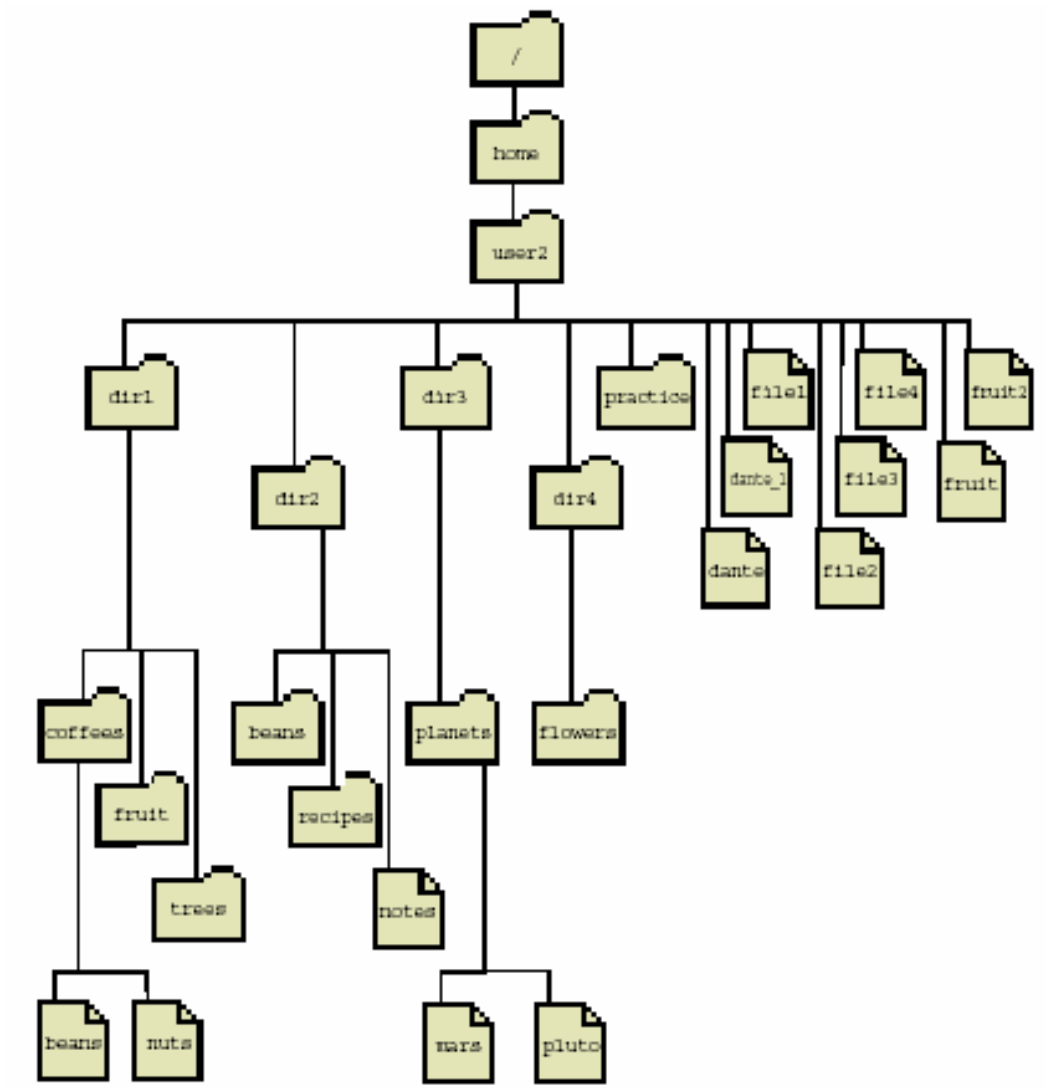
- Under Linux, nearly everything is represented as a file. Most physical devices in the system are accessed using special files in the file system

Directory Hierarchy

PRONET

VNU-ITP

UNDP



Standard Directories

PRONET VNU-ITP UNDP

`/`	The top of the Linux file system hierarchy
`/bin`	Essential command binaries which are required in single user mode.
`/boot`	Static files of the boot loader (contains everything required for the boot process except configuration files and the map installer)
`/dev`	Device files, contains the location of special or device files
`/etc`	Reserved for configuration files that are local to your machine (no binaries should be located under /etc)
`/etc/X11`	is the recommended location for all X11 host-specific configuration files
`/home`	User home directories
`/lib`	Essential shared libraries and kernel modules (contains those shared library images needed to boot the system and run the commands in the root file system)

Standard Directories

PRONET VNU-ITP UNDP

`/mnt`	mount point of *temporary* partitions. This is generally the location to where temporary file systems, such as floppies or CD-drives
`/opt`	reserved for the installation of add-on application software packages
`/root`	Home directory of root user
`/tmp`	temporary directory
`/etc/init.d`	Master startup scripts (not used as part of startup configuration)
`/etc/rc*.d`	Run-level startup configuration scripts, usually links to files in the `/etc/init.d` directory
`/usr/src`	source code
`/usr/include`	header files included by C programs
`/usr/lib`	libraries

Standard Directories

- Windows directories relate to Linux directories in terms of functionality

Windows	Linux
WinNT	/
system32	/bin, /sbin, /etc and /usr

- Do not confuse the top level root directory `/` with the home directory of the super user `/root`

FHS Data Types

PRONET

VNU-ITP

UNDP

- The File system Hierarchy System (FHS) defines two types of data use: *data sharing* and *data modification*. Each of these classifications has two opposing subtypes:

FHS Data Types

PRONET
VNU-ITP
UNDP

- **Data Sharing** : defines the type of data used in a network environment. Within data sharing, there are two subtypes:
  - Sharable
  - Non-Sharable
  - Data that is defined as *sharable* can be used by multiple users or by multiple hosts
  - *Non-sharable* data is data that is linked to a specific host. Passwords, configuration files, and logs are examples of non-sharable data

FHS Data Types

PRONET
VNU-ITP
UNDP

- Data Modification : defines how data can be augmented. There are two categories within this section :
  - Variable
  - Static
  - Variable refers to data that is changed by natural, frequent, processes
  - Static data is just that: data that doesnot change on a frequent or regular basis. Binary programs are an example of static data types

The Linux File System

- Linux supports disk partitioning:
  - One branch of directory structure can reside on one partition
  - Different types of file systems or mount options can be assigned to each partition

The Linux File System

- All UNIX-like file system types follow similar model:
  - Each file system on a partition (or slice) has an inode table
  - Inode table comprises *one record for each file* stored within this partition
  - Its inode number uniquely identifies a file within the file system

The Linux File System

- Each file has an inode table entry:
  - Inode table entry holds all attributes (meta data) of a file, such as: file size, user, group, permissions, etc.
- Directories map names into inode numbers:
  - Directories store links to inodes (and some/most store short symlinks on ext2 and ext3 file systems)
  - An inode number can have more than one link referencing it

Access Control

PRONET

VNU-ITP

UNDP

- Primary function: ***prevent unauthorized access*** to system data by automatically protect file and directory access by placing a standard set of ***access permissions*** when files and directories are created

Viewing File and Directory Permissions

PRONET VNU-ITP UNDP

- View *permissions* on files and directories by using the `ls -l` command.

- Example

```
$ ls -l .profile
```

Viewing File and Directory Permissions

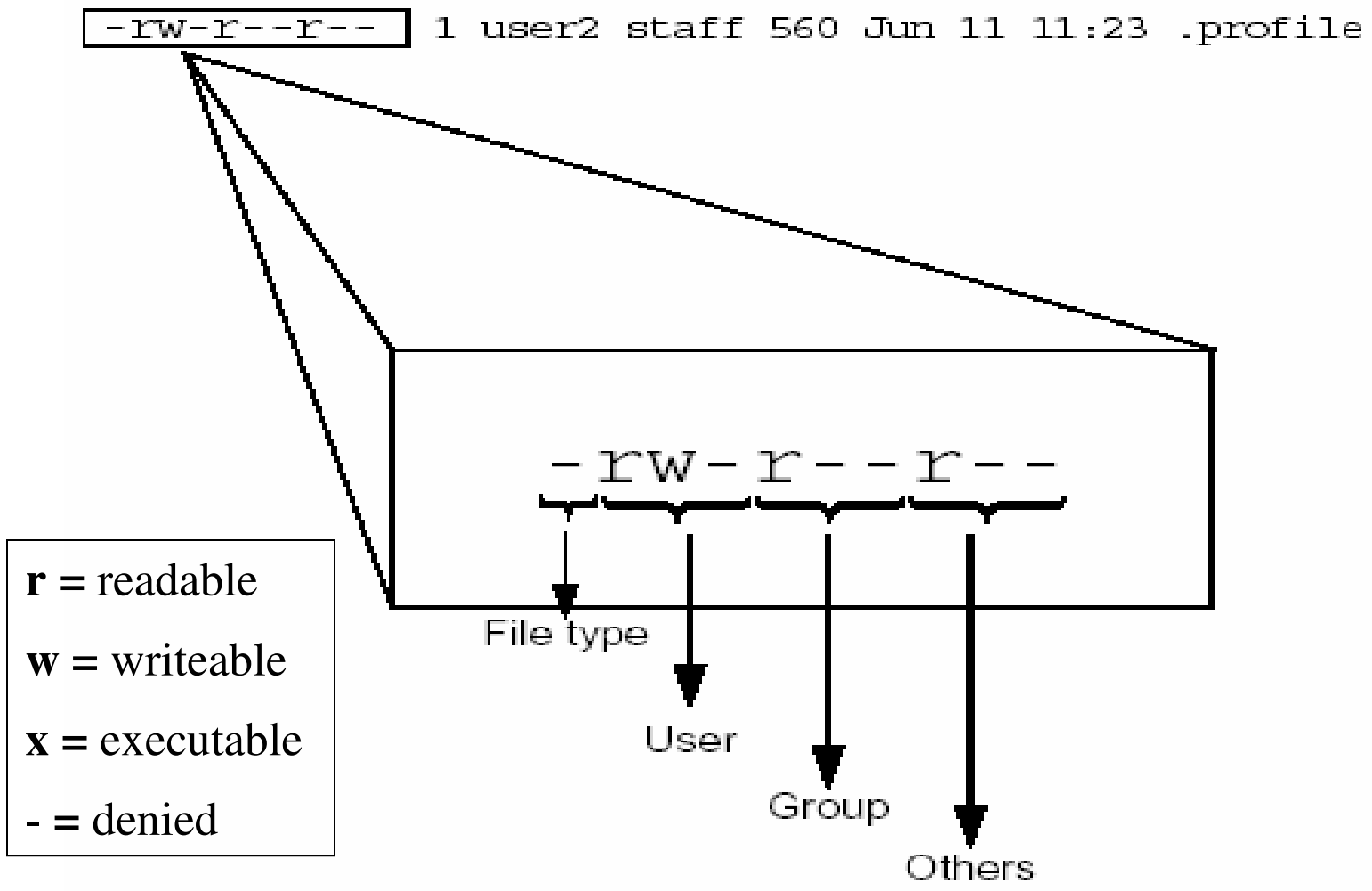


Figure 8-1 Permissions for Each Class of User

Permission Categories

PRONET

VNU-ITP

UNDP

- User (owner) permissions
- Group
- Others (world)

Determining Access to a File or Directory

PRONET VNU-ITP UNDP

- Access to a file or a directory is determined by the UID and the GID.
  - UID – Identifies the user who created the directory or file and determines ownership.
  - GID – Identifies the group of users who own the directory or file. A file or directory can belong to only one group at a time.
- To view these UID and GID numbers, use the

```
ls -n
```

Process For Determining Permissions

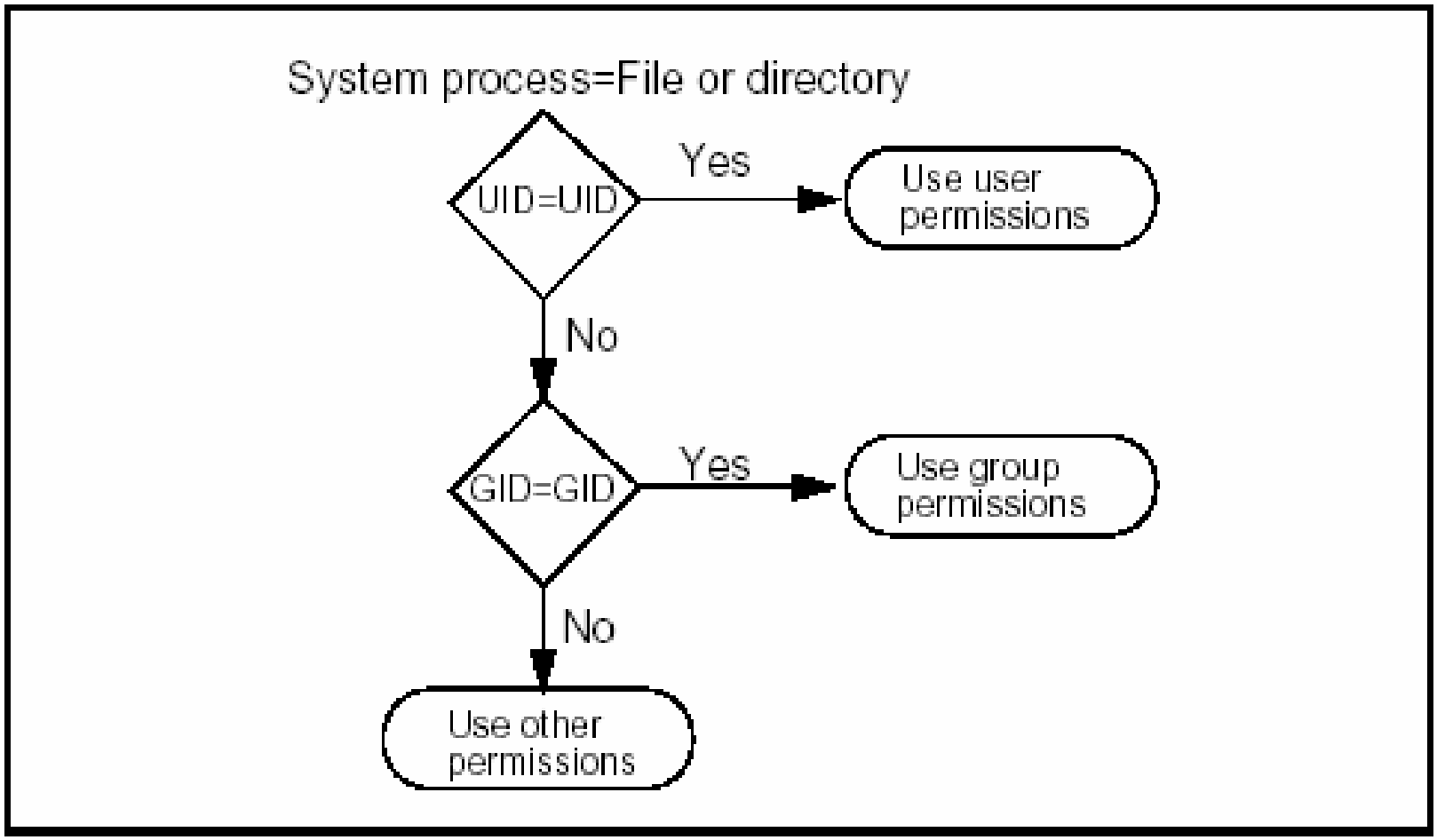


Figure 8-2 Process for Determining Permissions

Types of Permissions

- Permissions control ***who*** can do ***what*** to a file or directory and are represented by the characters:

r (read)

w (write)

x (execute)

- (denied)

Types of Permissions

Table 8-1 Permissions and Corresponding Symbols

Permission	Permission Symbol	Plain File	Directory
Read	`r`	File can be displayed or copied.	Contents can be listed with the `ls` command.[a]
Write	`w`	File contents can be modified.	Files can be added or deleted.[b]
Execute	`x`	File can be executed (shell scripts or executables only).	Access to the directory is controlled.[c]

a. To display a long listing (`ls -l`), you must also have execute (access) permission on the directory.

b. To add or delete files, you must have execute permission on the directory.

c. To copy a file from a directory, you must have execute permission on the directory. To use the `mv` command to place a new file in a directory or move a file from a directory, you must also have execute permission on the directory. You must also have read permission on the file itself for either of these actions to be performed.

Example 1

PRONET VNU-ITP UNDP

- **`-rwx------`**

  File is ***read/write/execute*** for **owner** only

- **`dr-xr-x---`**

  Directory is ***read/execute*** for **owner** and **group**

- **`-rwxr-xr-x`**

  File is ***read/write/execute*** for **owner**, and ***read/execute*** for **group** members **and others**

Example 2

PRONET VNU-ITP UNDP

- **`-rwxrw----`**

  File is ***read/write/execute*** for **owner** and ***read/write*** for **group**

- **`drwxr-x--x`**

  Directory is ***read/write/execute*** for **owner,** ***read/execute*** for **group** and ***execute*** for **others**

- **`dr-x-w-r--`**

  Directory is ***read/execute*** for **owner**, and ***write*** for **group** members and ***read*** for **others**

Changing Permissions

PRONET

VNU-ITP

UNDP

- You can modify the permissions set on files or directories using the **`chmod`** command.

$ **`chmod mode filename`**

- Either the **owner** of the file or directory or **superuser** can use this command to change permissions.

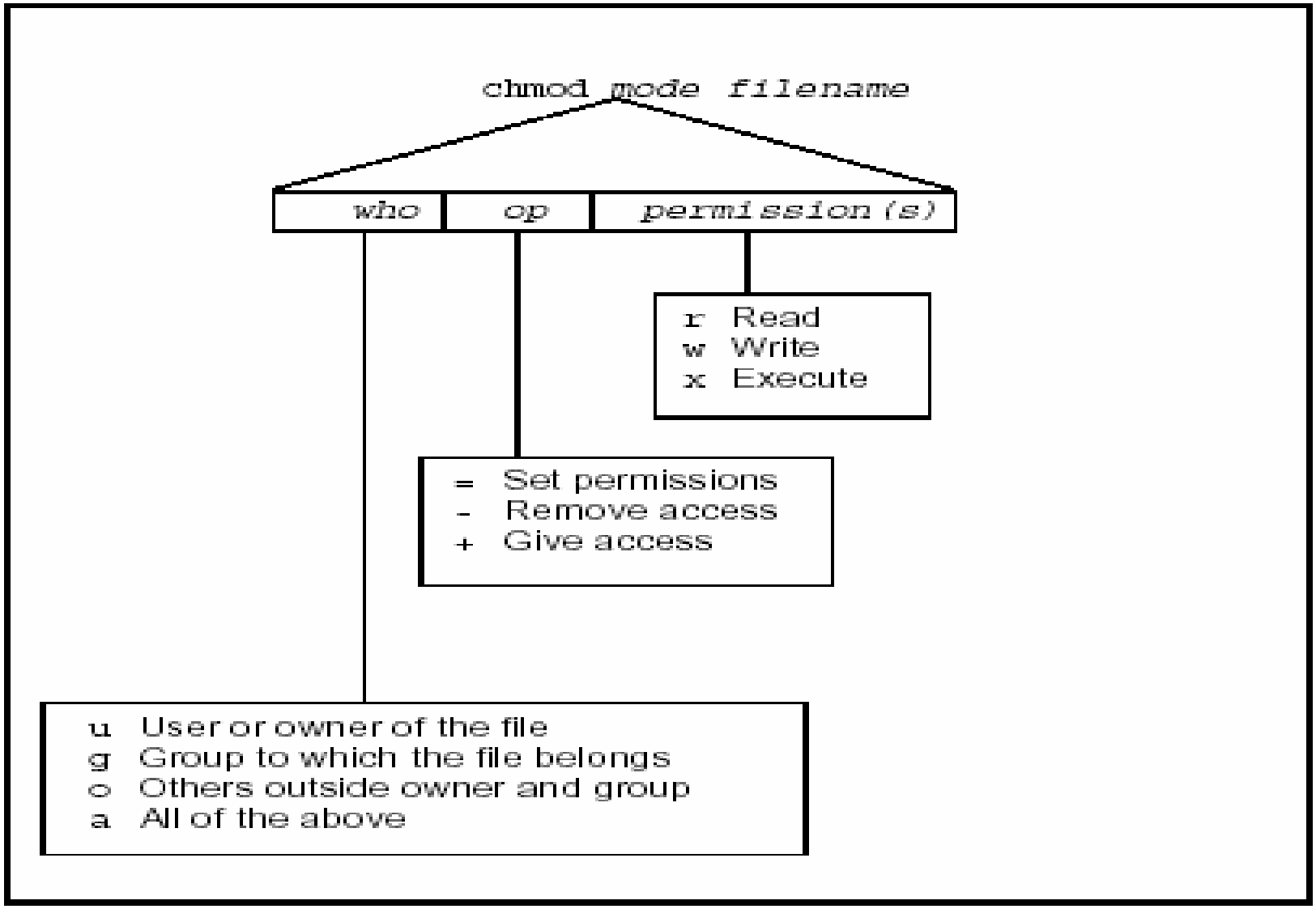
- The **`chmod`** command can modify permissions specified in either *symbolic* mode or *octal* mode.

Symbolic Mode

PRONET

VNU-ITP

UNDP



Example 1

PRONET VNU-ITP UNDP

- Remove **group *read*** permission :

```
$ ls -l dante
-rw-r--r-- 1 user2 staff 2 Jun 11 1:44 dante

$ chmod g-r dante
$ ls -l dante
-rw----r-- 1 user2 staff 2 Jun 11 1:44 dante
```

Example 2

PRONET VNU-ITP UNDP

- Add ***execute*** permission for **owner**, and ***read*** permission for **group** and **others**:

```
$ ls -l dante
-rw-------- 1 user2 staff 2 Jun 11 1:44 dante

$ chmod u+x,go+r dante
$ ls -l dante
-rwxr--r-- 1 user2 staff 2 Jun 11 1:44 dante
```

Example 3

- Set permission to ***read*** and ***write*** for **everyone**:

```
$ chmod a=rw dante
$ ls -l dante
-rw-rw-rw- 1 user2 staff 2 Jun 11 1:44 dante
```

Octal (Absolute) Mode

PRONET

VNU-ITP

UNDP

Each permission has an ***octal*** value **:**

Octal Value	Permissions
4	Read
2	Write
1	Execute

Octal Digits for Permission Sets

PRONET
VNU-ITP
UNDP

Octal Value	Permission Sets
7	r w x
6	r w -
5	r - x
4	r - -
3	- w x
2	- w -
1	- - x
0	- - -

Combined Values and Permissions

Command format :

$ **chmod *octal_mode filename***

Octal Mode	Permissions
644	rw-r--r--
751	rwxr-x--x
775	rwxrwxr-x
777	rwxrwxrwx

Example

PRONET

VNU-ITP

UNDP

- Give **user**, **group**, and **others** ***read*** and ***execute*** access :

```
$ ls -l dante
-rw-rw-rw- 1 user2 staff 2 Jun 11 11:54 dante

$ chmod 555 dante
$ ls -l dante
-r-xr-xr-x 1 user2 staff 2 Jun 11 11:54 dante
```

Compare two Mode

- Some symbolic mode expressions have no equivalent expression in absolute mode.
  - For example, `chmod u+x,g+w somefile` has no parallel in absolute mode.
- Absolute mode expressions are sometimes more concise than symbolic mode expressions.
- Absolute mode expressions are sometimes more suitable for use within shell scripts that might take input from other utilities in numeric form.

Default Permissions

PRONET

VNU-ITP

UNDP

- Default permissions, which are automatically assigned when a file or directory is created.
- The initial default permission value specified by the system for a file creation is 666 (rw-rw-rw-).
- The initial default permission value specified by the system for a directory creation is 777 (rwxrwxrwx).

The `umask` Filter

- The **`umask`** filter controls the default permissions assigned to newly created files and directories.
- The **`umask`** filter is a three-digit octal value that refers to ***read/write/execute*** permissions for **owner**, **group**, and **other**.
- The default value of **`umask`** is : **022**

Calculating How the File Mode Creation Mask is Applied

- Write the default permissions in their expanded (bitwise) form.
- Write the file mode creation mask beneath the default permissions.
- Perform the bitwise subtraction, and write down the result.

Example 1

PRONET VNU-ITP UNDP

- Default file permission **r w – r w – r w –** **(666)**

  umask of 022 **– – – – w – – w –** **(022)**

  Resulting file permission **r w – r – – r – –** **(644)**

- Default dir permission **r w x r w x r w x** **(777)**

  umask of 022 **– – – – w – – w –** **(022)**

  Resulting dir permission **r w x r – x r – x** **(755)**

Example 2 : Important note

PRONET VNU-ITP UNDP

- Default file permission **r w – r w – r w –** **(666)**

  umask of 123 **– – x – w – – w x** **(123)**

  Resulting file permission **r w – r – – r – –** **(644)**

  (*not* 543!)

- Default dir permission **r w x r w x r w x** **(777)**

  umask of 123 **– – x – w – – w x** **(123)**

  Resulting dir permission **r w – r – x r – –** **(654)**

Changing the *umask* Value

PRONET VNU-ITP UNDP

- Command format :

umask [*new_value*]

(umask will be valid for current shell and subshells.)

- Example :

1. Verify the current umask.

```
$ umask
022
```

2. Change the umask value to 027 and verify.

```
$ umask 027
$ umask
027
```

File & Dir Permission with umask = 027?

PRONET VNU-ITP UNDP

- Default file permission **r w – r w – r w –** **(666)**

  umask of 027 **– – – – w – r w x** **(027)**

  Resulting file permission **r w – r – – – – –** **(640)**

- Default dir permission **r w x r w x r w x** **(777)**

  umask of 027 **– – – – w – r w x** **(027)**

  Resulting dir permission **r w x r – x – – –** **(750)**

Special Permissions

- Three types of permission are available for executable files and public directory:
  - Set user ID : `suid`
  - Set group ID : `sgid`
  - Sticky bit

Set User ID

- When `suid` is set on executale files, a user or process that runs this file *is granted access based on the owner of the file* (usually `root`) instead of user who started the file
- The `suid` permission displays as an “s” in the owner’s executable field. If file is not executable, `ls` shows capital “S”

`# ls -l /bin/su /usr/bin/passwd`

```
-rwsr-xr-x 1 root root 18452 Jul 2 2003 /bin/su
-r-s--x--x 1 root root 13476 Aug 7 2003 /usr/bin/passwd
```

Set User ID

- The `root` user and owner can set the suid permission on file using `chmod` command and the octal value *4000* or symbolic *s*:

```
# chmod 4755 <executable_file>

# chmod u+s <executable_file>
```

Set Group ID

PRONET

VNU-ITP

UNDP

- When `sgid` is similar to `suid`, except that a user or process that runs this file *is granted access based on the owner's group of the file*
- The `sgid` permission displays as an "s" in the group's executable field. If file is not executable, `ls` shows capital "S"

`# ls -l /usr/bin/slocate /usr/bin/write`

```
-rwxr-sr-x 1 root slocate 9 Jul 2 2003 /usr/bin/slocate
-rwxr-sr-x 1 root tty 13476 Aug 7 2003 /usr/bin/write
```

Set Group ID

- The `sgid` is a useful feature for creating *shared directories*: files created in these directories belong to the group to which the directories belong
- The `root` user and owner can set the sgid permission on file using `chmod` command and the octal value *2000* or symbolic *s*:

```
# chmod 2755 <executable_file>

# chmod g+s <executable_file>
```

Sticky Bit Permission

PRONET

VNU-ITP

UNDP

- It protects the files in within a public writable directory. If the directory has the sticky bit set, then:
  - Only the owner and root can delete files
  - Owner still need write permission to the directory
- The `sticky bit` permission displays as an “t” in the *other’s* executable field. If file is not executable, `ls` shows capital “T”

```
# ls -ld /tmp
drwxrwxrwt 8 root root 4096 Jul 2 2003 /tmp
```

chown Command

- You use `chown` command to change the origin owner of a file or directory to another user on the system

```
# ls -l /usr/bin/slocate
-rwxr-sr-x 1 root slocate 9 Jul 2 2003 /usr/bin/slocate
# chown minh /usr/bin/slocate
# ls -l /usr/bin/slocate
-rwxr-sr-x 1 minh slocate 9 Jul 2 2003 /usr/bin/slocate
```

chgrp Command

- Use the `chgrp` command to change the group of the files or directories to another group on the system

```
# ls -l /usr/bin/slocate
-rwxr-sr-x 1 root slocate 9 Jul 2 2003 /usr/bin/slocate
# chgrp instructors /usr/bin/slocate
# ls -l /usr/bin/slocate
-rwxr-sr-x 1 root instructors 9 Jul 2 2003 /usr/bin/slocate
```

Command User and Group Ownership Simultaneously

PRONET

VNU-ITP

UNDP

- `chown` command help you to change owner and group of a file or directory simultaneously

```
# ls -l /usr/bin/slocate
-rwxr-sr-x 1 root slocate 9 Jul 2 2003 /usr/bin/slocate
# chown minh:instructors /usr/bin/slocate
# ls -l /usr/bin/slocate
-rwxr-sr-x 1 minh instructors 9 Jul 2 2003 /usr/bin/slocate
```

chown and chgrp Commands

- You can also ownership and group recursively with –R option:

```
chown –R <user> <file(s)>

chgrp –R <group> <file(s)>
```

`chattr` Command

- The `chattr` command changes file attributes on an ext2 or ext3 file system. Using the different options, `chattr` can mark files as *immutable*, *secure deletion*, and more. The syntax for `chattr`:

```
chattr <options> <mode files>
```

- To assign options, `chattr` uses three different options (called opcodes):
  - **+** Add attribute
  - **-** Remove attribute
  - **=** Assign attributes (removing unspecified attributes)

`chattr` Command

PRONET
VNU-ITP
UNDP

- The following is a list of *attributes* used with chattr:

`a`	Append only for writing. Can be set or cleared only by a privileged user
`c`	Compressed
`d`	No dump
`i`	Immutable. Can be set or cleared only by a privileged user.
`s`	Secure deletion; the contents are zeroed on deletion
`S`	Synchronous updates

`lsattr` Command

- You can display attributes of a file are set by `chattr` by using this command

Linking Files and Directories

- *Links* are used to create **alternate names** or **aliases** for files and directories on a system

- There are two kinds of links:
  - ***hard* link**
  - ***symbolic* link (or *soft* link)**

Hard link

PRONET

VNU-ITP

UNDP

- Used to link files on the **same file system**
- Files that are hard linked **share the same inode** number (refer to the same data on disk)
- Hard links are **not used to link directories** and **cannot cross file systems.**

Hard link

PRONET

VNU-ITP

UNDP

- Hard link count can be displayed using ls –l command

```
$ ls -l ~
-rw-------  1 torey staff 368  Oct 19 dante_1
drwx--x--x  5 torey staff 512  Oct 19 dir1
drwx--x--x  4 torey staff 512  Oct 19 dir2
```

- The link count on directories includes a link to the current directory (.) and from the parent (..) directory, and a number of subdirectory included in the directory.

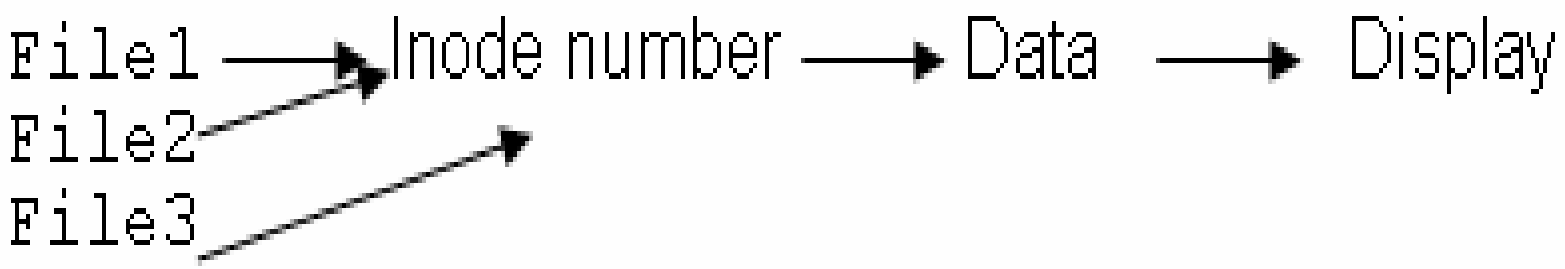
Hard link

PRONET

VNU-ITP

UNDP

The structure of a hard link is as follows:



Symbolic Link

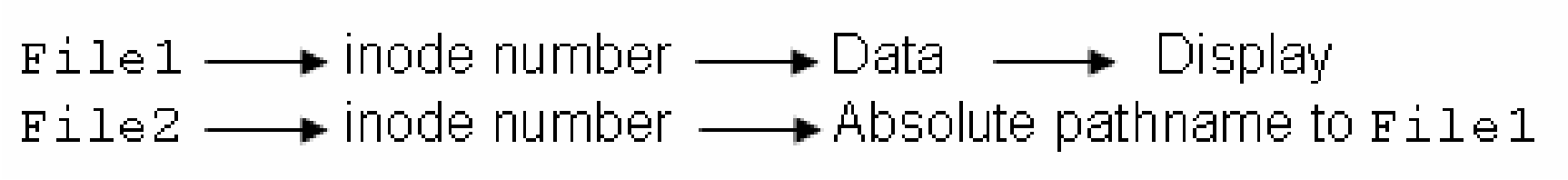
PRONET VNU-ITP UNDP

- **Used to link files or directories.**
- **Symbolically linked files do not share a single inode, these links can cross file system .**
- **A file with symbolically link:**

```
$ ls -l Test
-rw-r--r-- 1 torey staff 35 May 8 linktest
lrwxrwxrwx 1 torey staff 8 May 12 symlink--> linktest
$ ls -F
linktest
symlink@
```

Symbolic Link

The structure of a symbolic link is as follows:



Creating Link

PRONET VNU-ITP UNDP

- Command Format :

  ln [-s] <*source_file*> <*target_file*>

Examle – Creating Hard Link

PRONET

VNU-ITP

UNDP

```
$ ln /export/home/user2/dante essay

$ ls -i /export/home/user2/dante
89532 dante

$ ls -i essay
89532 essay
```

Examle – Creating Symbolic Link

PRONET

VNU-ITP

UNDP

```
$ ln -s tutor.vi symlink

$ ls -l symlink

lrwxrwxrwx 1 torey staff 8 May 9 symlink--->tutor.vi
```

Summary

PRONET

VNU-ITP

UNDP

- Describe and explain the Linux file system directory structure
- Explain file system concepts
- Identify and explain inodes
- Utilize `chown`, `chgrp`, and other file related commands
- Set user and group ID permissions
- Identify and explain sticky bits
- Identify and explain links