Ejercicios prácticas Sistemas Operativos

Alejandro Rubio Martínez

Módulo I

Sesión 1

Actividad 1.1

El script para ejecuatr el UML suponiendo que los archivos necesarios se encuentran en una carpeta llamada fedora al mismo nivel que el script (necesaria orden `gunzip`):

```
BASE_DIR="fedora"
TMP_DIR="/tmp/uml"


if ! [ -d $TMP_DIR ]; then
  mkdir $TMP_DIR
fi

if [ -f "$BASE_DIR/kernel32-3.0.4.gz" ]; then
    gunzip -k "$BASE_DIR/kernel32-3.0.4.gz"
else
    if ! [ -f "$BASE_DIR/kernel32-3.0.4" ]; then
        echo "ERROR: Falta el archivo del kernel"
        rm -rf $TMP_DIR
        exit 1
    fi
fi
mv "$BASE_DIR/kernel32-3.0.4" $TMP_DIR

if [ -f "$BASE_DIR/Fedora14-x86-root_fs.gz" ]; then
    gunzip -k "$BASE_DIR/Fedora14-x86-root_fs.gz"
else
    if ! [ -f "$BASE_DIR/Fedora14-x86-root_fs" ]; then
        echo "ERROR: Falta el archivo de fedora root"
        rm -rf $TMP_DIR
        exit 1
    fi
fi
mv "$BASE_DIR/Fedora14-x86-root_fs" $TMP_DIR

cp "$BASE_DIR/kernel32-3.0.rar" $TMP_DIR

chmod u+x $TMP_DIR/kernel32-3.0.4
exec $TMP_DIR/kernel32-3.0.4 ubda=$TMP_DIR/Fedora14-x86-root_fs mem=1024m
```

Actividad 1.2

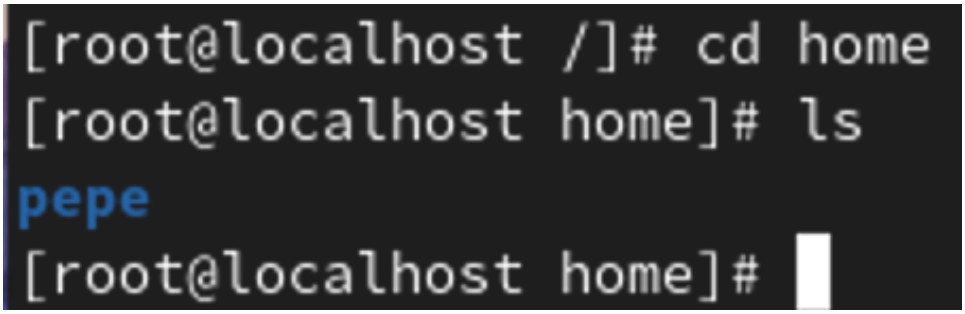
El contenido de /etc/default/useradd es:

```
# useradd defaults file
GROUP=500
HOME=/home
INACTIVE=-1
EXPIRE=
SHELL=/bin/bash
SKEL=/etc/skel
CREATE_MAIL_SPOOL=yes
```

El contenido de /etc/login.defs (uncommented) es:

```
#QMAIL_DIR  Maildir
MAIL_DIR  /var/spool/mail
#MAIL_FILE  .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

#USERDEL_CMD  /usr/sbin/userdel_local

CREATE_HOME yes

UMASK           077

USERGROUPS_ENAB yes

ENCRYPT_METHOD SHA512
```

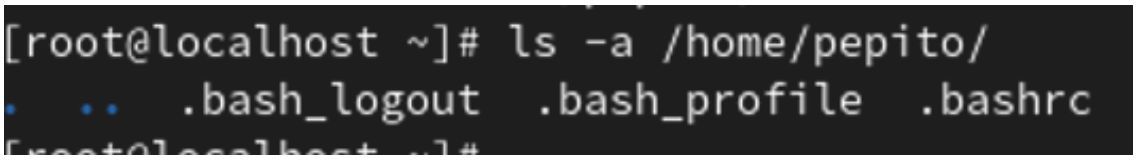
Para crear un nuevo usuario utilizamos el comando `useradd <nombre>`. Vemos entonces que en el fichero /etc/passwd ha aparecido la línea `pepe:x:500:500::/home/pepe:/bin/bash`, y en el fichero /etc/group la línea `pepe:x:500:`. Ahora veamos el /home:



Podemos observar como todo ha salido según lo esperado.

Actividad 1.3

1. Se crean tres usuarios, el primero se le borrará su cuenta en un día (cambiar la línea `PASS_MAX_DAYS` en /etc/login.defs), el segundo no dispondrá de `/home` (cambiar la línea `CREATE_HOME` esta vez) y el tercero con los parámetros estándares.
2. Se utiliza el comando `userdel` para borrar los dos primeros y podemos observar como el primero sólo deja su directorio `/home`, mientras que el del segundo se especificó que no se creara.
3. Accedemos al tercer usuario y utilizamos `la -a /home` para ver lo siguiente:

   Podemos ver como los únicos archivos que existen se encuentran ocultos y son los archivos básicos para la configuración del bash.

Actividad 1.4

Por orden veremos el nombre de usuario, una x que marca que la contraseña se encuentra en /etc/passwd, el UID (ID del usuario), el GID (ID del grupo), información del ID del usuario, la ruta del directorio de inicio y la ruta de un comando o shell asociado.

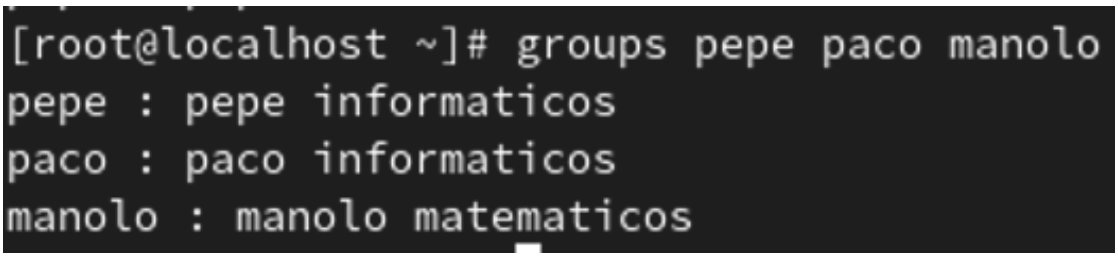
Actividad 1.5

El archivo /etc/shadow contiene información sobre la contraseñas de los usuarios, por lo que solo tendrá permiso para verlo el root, por lo que entonces al intentar visualizarlo con otro usuario da error de permisos.

Actividad 1.6

1. Voy a empezar creando tres usuarios distintos pepe, paco, manolo (`useradd <nombre>`), y dos grupos (`groupadd <nombre>`) llamados informáticos y matemáticos.

   Ahora utilizamos la orden `gpasswd -a <usuario> <grupo>` para añadir pepe y paco a informaticos y manolo a matemáticos. Con la orden `groups <usuarios>` podemos ver a que grupo pertenecen:

2. Veamos lo que ocurre:

   Podemos observar que primero nos dará el id de usuario (0 por ser el root), y luego todos los grupos a los que perteence.

Actividad 1.7

Utilizamos `find / kernel32-3.0.4.gz`

/home/parallels/Desktop/Parallels Shared Folders/Home/Desktop/ugr/So/fedora/kernel32-3.0.4.gz

Actividad 1.8

En la carpeta /var/tmp en la que el tiempo que se mantienen los archivos depende de la distro de la que usemos.

Actividad 1.9

Visualicemos el archivo /etc/fstab:

```
# /etc/fstab
#
LABEL=ROOT          /    auto  noatime 1 1
tmpfs            /dev/shm  tmpfs defaults 0 0
tmp             /tmp     tmpfs rw,mode=1777,fscontext=system_u:object_r:tmp_t:s0 0 0
devpts             /dev/pts  devpts  gid=5,mode=620  0 0
sysfs            /sys     sysfs defaults 0 0
proc             /proc    proc  defaults 0 0
```

Ahora el archivo /etc/mtab:

```
LABEL=ROOT / auto rw,noatime 0 0
proc /proc proc rw 0 0
sysfs /sys sysfs rw 0 0
devpts /dev/pts devpts rw,gid=5,mode=620 0 0
tmpfs /dev/shm tmpfs rw 0 0
/tmp /tmp tmpfs rw,mode=1777 0 0
none /proc/sys/fs/binfmt_misc binfmt_misc rw 0 0
```

La única diferencia entre ambos fichero es que fstab muestra los sistemas de archivos que se montarán cuando se arranque el sistema mientras que mtab muestra los montados actualmente.

Actividad 1.10

Por orden lo que podemos ver en cada línea es:

El dispositivo en el que se encuentra montado, el punto de montaje, el tipo de sistemas de archivos, las opciones, que son las siguientes:

- **async**: las escrituras al filesystem se demoran, es decir que no se hacen en el momento sino que se

hacen varias escrituras juntas. Esto da un mayor rendimiento, aunque si el sistema se cuelga, apaga o el filesystem se desmonta, los datos se pederán si aún no han sido escritos.
- **auto**: el sistema de archivos será montado automáticamente al iniciar el sistema o al ejecutar el comando mount -a.
- **noauto**: debe ser montado explícitamente.
- **defaults**: utiliza las opciones por defecto, que son rw, suid, dev, exec, auto, nouser, async.
- **ro**: monta el filesystem como de sólo lectura.
- **rw**: monta el filesystem como lectura/escritura.
- **user**: permite que cualquier usuario pueda montar el filesystem.
- **nouser**: especifica que el filesystem sólo puede ser montado por el usuario root y no por un usuario común.
- **sync**: todas las escrituras al filesystem se hacen en el momento. Esto da mayor seguridad a los datos pero un menor rendimiento.

Las últimas dos columnas indican si se tiene que hacer backup del sistema (1 si, 0 no) y la última si debe ser checkeado (0 no, 1 sí con prioridad y 2 sí sin prioridad).

Actividad 1.11

En la actividad 1.9 ya se ha explicado qué son /etc/fstab y /etc/mtab y sus diferencias.

Ahora veamos que contiene el archivo /proc/filesystems:

```
nodev sysfs
nodev rootfs
nodev bdev
nodev proc
nodev cgroup
nodev cpuset
nodev tmpfs
nodev devtmpfs
nodev binfmt_misc
nodev securityfs
nodev sockfs
nodev pipefs
nodev anon_inodefs
nodev rpc_pipefs
nodev configfs
nodev devpts
  reiserfs
  ext3
  ext2
  ext4
  squashfs
nodev ramfs
  vfat
  msdos
  iso9660
nodev ecryptfs
nodev nfs
```

```
nodev nfs4
nodev nfsd
nodev cifs
  ntfs
nodev autofs
nodev fuse
  fuseblk
nodev fusectl
  udf
  nilfs2
nodev hostfs
  btrfs
  gfs2
  gfs2meta
nodev mqueue
nodev selinuxfs
```

Ahora /proc/mounts:

```
rootfs / rootfs rw 0 0
/dev/root / ext4 rw,noatime,user_xattr,acl,barrier=1,data=ordered 0 0
none /proc proc rw,nosuid,nodev,noexec,relatime 0 0
none /sys sysfs rw,nosuid,nodev,noexec,relatime 0 0
devpts /dev/pts devpts rw,relatime,gid=5,mode=620 0 0
/tmp /tmp tmpfs rw,relatime 0 0
none /proc/sys/fs/binfmt_misc binfmt_misc rw,relatime 0 0
```

La diferencia entre estos archivos es que /proc/filesystems indica los sistemas de archivos soportados por el kernel que estemos usando mientras que /proc/mounts es una representación en forma de archivo de los dispositivos disponibles por el kernel de Linux. Este archivo es bastante parecido a /etc/mtab, pero incluye otros sistemas de archivos que no tienen nada que ver con discos duros.