P09-Las propiedades de protección de los ficheros

¿Qué hace a un fichero ejecutable?¿Qué derechos tienen los usuarios sobre el uso de los ficheros?¿Hay permisos especiales?

Bibliografía

- [Capítulos 2,3 y 4] F.M. Márquez: UNIX. Programación Avanzada 3ª Edición. Rama, 2004.
- [Apartados 5.2 y 5.3] C. Rodríguez, I. Alegria, J. González, A. Lafuente: *Descripción Funcional de los Sistemas Operativos*. Síntesis, 1994
- [Capítulo 12] W. Stallings: Sistemas Operativos. 5º Ed. Pearson Prentice-Hall, 2005.
- [Capítulo 13] G. Nutt: Sistemas Operativos. 3º Ed. Pearson Addison Wesley, 2001
- [Capítulo 12] A. Silberschartz: Operating Systemas Concepts. Sixth, John Wiley & Son, 2003

Introducción

- Uso compartido de un computador entre diferentes usuarios
  - Simultáneo o alternado
- El sistema operativo necesita identificar a los usuarios que lo están utilizando
  - Necesidad de un control de acceso (login / logout), basado en una identidad de usuario y contraseña
- El sistema operativo debe proporcionar mecanismos de seguridad y/o protección en el acceso a la información perteneciente a los usuarios
  - Información almacenada en el sistema de ficheros
    - Acceso controlado al sistema de ficheros
  - Información relativa a los procesos que ejecutan los usuarios (Proyectos 10-14)

Sistemas multiusuario.

Accounting: Contabilidad

Confidencialidad

Seguridad

Gestión de privilegios

Comprobar accesos

Contabilizar el uso de los recursos

Limitar el uso de los recursos (cuotas)

Sistemas multiusuario.

- Dominios de protección:
  - Un sistema de computo se compone de objetos software y hardware
  - Los proceso tendrán acceso a ciertas operaciones sobre ciertos objetos
  - Un dominio de protección define un cjto de objetos y operaciones
    - Cjo de parejas (Objeto, operaciones)
  - Un proceso actúa dentro de un dominio de protección
  - Relación Proceso – dominio puede ser estática o dinámica
  - Matrices de acceso (Dominio - Recurso/Objeto- Operación)
- Dominios en UNIX
  - Dominio asociado al usuario
- Tipos de usuarios.
  - (Privilegiado, normal, otros..)
- Grupos de usuarios
  - Diferentes roles
- Propietario y grupo de un recurso

	Rec1	Rec2	Rec3	Rec k	Rec n
Dom 1			RW		RX
Dom 2	R	RW			
Dom j			R	RW	RWX

4.3 Mecanismos de Protección.

Privilegios:
- Ninguno
- Conocimiento
- Ejecución
- Lectura
- Adición
- Actualización
- Cambio Protección
- Borrado
- …

Privilegios UNIX:
- Lectura
- Escritura
- Ejecución

propietario	grupo	resto
RWX	RWX	RWX

Información de protección se encuentra en el I-NODE

Estructura del i-node (UNIX)

i-node

Tipo
Permisos
Propietario
Grupo
Tamaño
Fechas
Nºenlaces
...

Índices de Bloques Datos

Tipo de fichero: Ordinario, Directorio, caracteres, de bloques, Pipe, Enlace, Shocket

Permisos de RWX para el propietario, Grupo o resto de usuarios

Identificación de Propietario y Grupo,

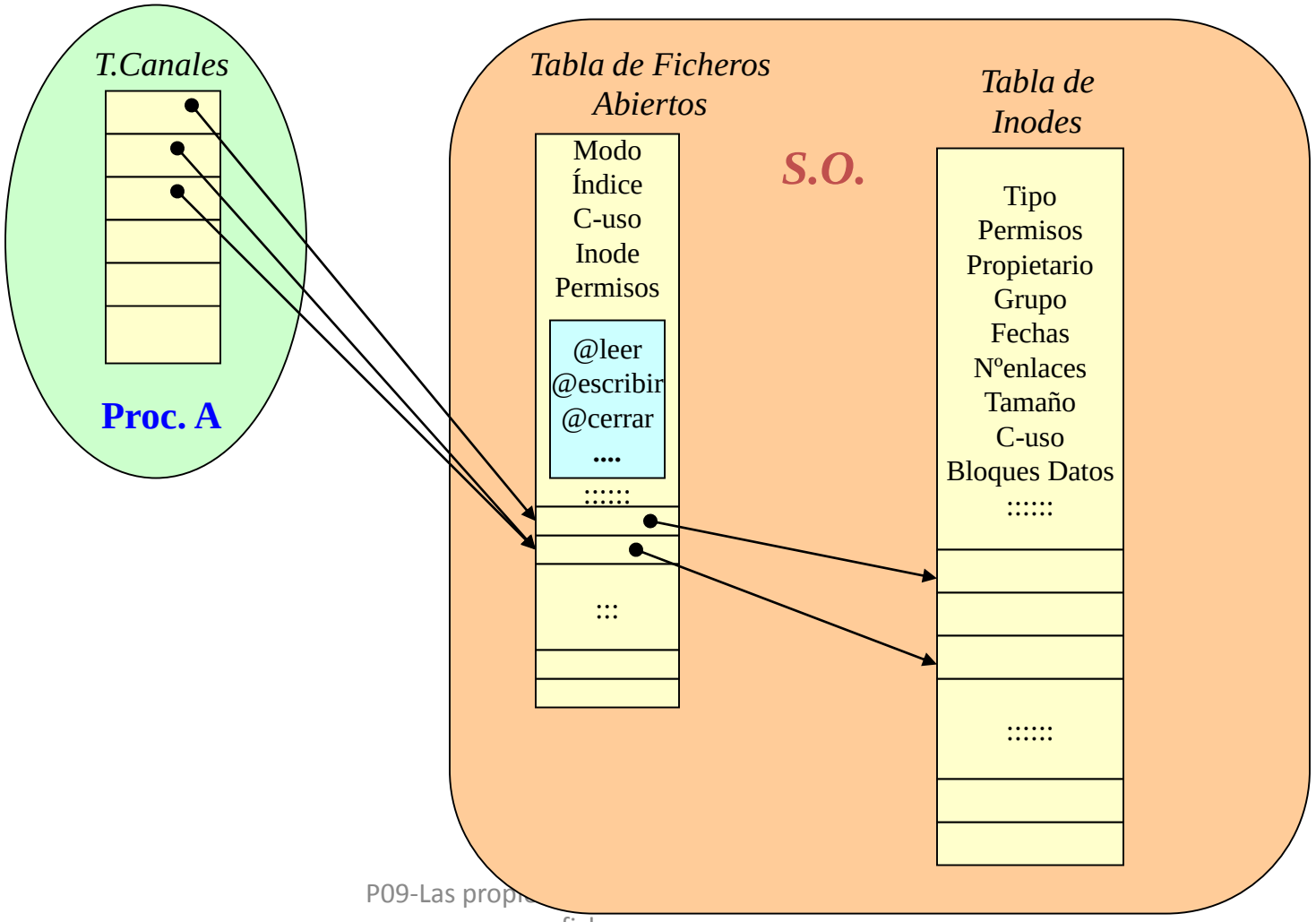
Tamaño en bytes del fichero

Varias fechas como la del último acceso o la de la última modificación de datos.

Nº de enlaces al fichero (nº de nombres en el sistema de ficheros)

Otros datos como número de dispositivo, stiky-Bit (cod. Reentrante), Cambiar UID o GIG.

Protección **se controla en la apertura** del dispositivo/fichero



Llamadas al Sistema para gestión del sistema de ficheros (UNIX/Linux)

- **Apertura/lectura/escritura/ubicación**
  - `int open (char *camino, int flags, mode_t perm);`
  - `int open (char *camino, int flags);`
  - `int creat (char *camino, mode_t perm);`

Flags:

O_RDONLY	**O_WRONLY**	**O_RDWR**	O_NDELAY
O_APPEND	O_DSYNC	O_RSYNC	O_SYNC
O_NOCTTY	**O_CREAT**	O_EXCL	O_TRUNC

- **Control de ficheros-directorios**
  - `int mkdir (char *path, mode_t mode);`

Llamadas al Sistema (UNIX) cont.I

- **Ficheros y control de dispositivos**
  - `int stat (char *path, struct stat *sbuf);`
  - `int fstat (int fd, struct stat *sbuf);`

Estructura *stat*

- **st_dev**: identificador del dispositivo que contiene el fichero (short)
- **st_ino**: número de inodo (ushort)
- **st_mode**: modo (short) bits de permisos
- **st_nlink**: número de enlaces (short)
- **st_uid**: identificador del dueño (ushort)
- **st_gid**: identificador de grupo (ushort)
- **st_rdev**: tipo de dispositivo para ficheros especiales (short)
- **st_size**: Tamaño en bytes (long) (0 en los ficheros especiales)
- **st_atime**: Hora del último acceso (long)
- **st_mtime**: Hora de la última modificación (long)
- **st_ctme**: Hora de creación o cambio de estado(long)
- **...**

Llamadas al Sistema (UNIX) cont.IV

- **Multiusuario**
  - `int chmod (char *path, mode_t modo);`
  - `int fchmod (int fd, mode_t modo);`
  - `int chown (char *path, int propietario, int grupo);`
  - `int access (char *path, int modo);`

    `Con modo: R_OK, W_OK, X_OK, F_OK`

  - `mode_t umask (mode_t modo); // 022 (máscara típica)`

  - `uid_t getuid(void); uid_t geteuid(void);`
  - `gid_t getgid(void); uid_t getegid(void);`

Mascaras de bits de ***mode_t***

Tipo	Especiales	propietario	grupo	resto
xxx	---	RWX	RWX	RWX

S_IRUSR (00400) read by owner
S_IWUSR (00200) write by owner
S_IXUSR (00100) execute/search by owner ("search" applies for directories, and means that entries within the directory can be accessed)

S_IRGRP (00040) read by group
S_IWGRP (00020) write by group
S_IXGRP (00010) execute/search by group

S_IROTH (00004) read by others
S_IWOTH (00002) write by others
S_IXOTH (00001) execute/search by others

Mascaras de bits de ***mode_t***

BITS ESPECIALES:

Tipo	***Especiales***	propietario	grupo	resto
xxx	***SST***	RWX	RWX	RWX

S_ISUID (04000) ***set-user-ID*** (set process effective user ID on execve(2))

S_ISGID (02000) ***set-group-ID*** (set process effective group ID on execve(2); mandatory locking, as described in fcntl(2); take a new file's group from parent directory, as described in chown(2) and mkdir(2))

S_ISVTX (01000) ***sticky bit*** (restricted deletion flag, as described in unlink(2))

Llamadas al Sistema (UNIX) cont.IV

- **Multiusuario**
  - `struct passwd *getpwnam(const char * nombre);`
  - `struct passwd *getpwuid(uid_t uid);`

```
struct passwd {
    char    *pw_name;       /* nombre de usuario */
    char    *pw_passwd;     /* contraseña cifrada */
    uid_t   pw_uid;         /* id. del usuario */
    gid_t   pw_gid;         /* id. del grupo primario */
    char    *pw_gecos;      /* nombre real */
    char    *pw_dir;        /* directorio de inicio */
    char    *pw_shell;      /* Shell por defector */
  };
```

```
#include <pwd.h>
#include <sys/types.h>
```

Llamadas al Sistema (UNIX) cont.IV

- **Multiusuario**
  - `struct group *getgrnam(const char *nombre);`
  - `struct group *getgrgid(gid_t gid);`

```
struct group {
    char    *gr_name;       /* nombre del grupo */
    char    *gr_passwd;     /* contraseña del grupo */
    gid_t   gr_gid;         /* ID del grupo */
    char    **gr_mem;       /* vector de apuntadores a todos los/*
                            /* nombres de miembros del grupo */
   };
```

```
#include <grp.h>
#include <sys/types.h>
```

Ejemplo: permisos.c

```
/* ========================================================================
 Name        : permisos.c
……

#define MENSAGE_ERROR1 "No argumentos erroneo\n"

int main(int argc, char *argv[])
{
  int j;
  struct stat infoinode;
  mode_t  modo;
  char Linea[256];
  struct passwd *ppas;
  struct group  *pgrp;

  if (argc < 2 )  {
    write (2, MENSAGE_ERROR1, strlen(MENSAGE_ERROR1));
    exit(1);
  }
  for (j=1; j< argc; j++)  {
    stat(argv[j],&infoinode);
    modo = infoinode.st_mode;
    sprintf(Linea, "%12s : Mode(%6o) ", argv[j], modo);
    ppas = getpwuid(infoinode.st_uid);
    pgrp = getgrgid(infoinode.st_gid);
    sprintf(Linea , "%s %s %s ", Linea, ppas->pw_name, pgrp->gr_name);
    traza_modo(modo, Linea+strlen(Linea));
    sprintf(Linea , "%s \n", Linea);
    write( 1, Linea, strlen(Linea));
  }
  return EXIT_SUCCESS;
}
```

Ejemplo: permisos.c

```
/*
============================================================================
 Name        : permisos.c
 Author      : G.A. & M.L.
 Version     : 1.0
 Copyright   : ATC-KAT - Fac. Informatica - UPV/EHU
 Description : Ejemplo de mostrar los permisos de un fichero
 ============================================================================
 */
#include <sys/types.h>
#include <sys/stat.h>
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <pwd.h>
#include <grp.h>
```

```
const mode_t vmodos[] = {
        S_IRUSR, S_IWUSR, S_IXUSR, S_IRGRP, S_IWGRP, S_IXGRP,
        S_IROTH, S_IWOTH, S_IXOTH
};

const char mode_chars[3]   = "rwx";

void traza_modo(mode_t modo, char *Linea)
{
  char car, *pLinea;
  int i;
  pLinea =Linea;
  for (i=0; i< 9; i++)  {
    if(vmodos[i] & modo) car= mode_chars[i%3];
    else car ='-';
    *pLinea = car;
    pLinea++;
  }
  *pLinea = '\0';
}
```

Leer el valor de máscara de usuario

```
mode_t read_umask()
{
        mode_t mask;

        mask = umask(0);

        traza_modo(modo, Linea+strlen(Linea));
        sprintf(Linea , "%s \n", Linea);
    write( 1, Linea, strlen(Linea));
        umask(mask);

        return mask;
}
```