Sistemas operativos II

Sistema de ficheros en UNIX

Estructura del sistema de ficheros de unix

Sistema de ficheros en UNIX

Estructura del sistema de ficheros de unix

Sistema de ficheros en UNIX

Estructura del sistema de ficheros de unix

Punto de vista del usuario

- estructura jerárquica
- posibilidad de crear y borrar ficheros
- crecimiento dinámico de los ficheros
- protección de los datos de los ficheros
- tratamiento de los dispositivos periféricos como ficheros
- cada fichero tiene un nombre completo que es una secuencia de nombres separados por el caracter /; cada uno de los nombres designa un nombre único en el nombre previo
- un fichero es una sucesión de bytes
- un directorio es un fichero normal
- permisos de acceso controlados (lectura, escritura, ejecución) rwxrwxrwx
- árbol con un nodo raiz (/):
  - cada nodo que no es una hoja es un directorio
  - cada hoja: fichero, directorio o dispositivo
- en realidad se trata de un grafo y no de un árbol: enlaces reales y simbólicos

Punto de vista del kernel

- un fichero es una sucesión de bytes
- un fichero está representado por una estructura pequeña, con la informacion que el kernel necesita conocer de dicho fichero, denominada inodo
  - propietario y grupo del fichero (*uid y gid*)
  - *modo* del fichero: entero codificado bit a bit con los permisos y el tipo de fichero
  - fechas (último acceso, última modificación, último cambio en el inodo)
  - tamaño
  - número de enlaces reales
  - direcciones de disco que ocupa

inodos en openBSD

```
struct  ufs1_dinode {
    u_int16_t       di_mode;        /*   0: IFMT, permissions; see below. */
    int16_t         di_nlink;       /*   2: File link count. */
    union {
            u_int16_t oldids[2];    /*   4: Ffs: old user and group ids. */
            u_int32_t inumber;      /*   4: Lfs: inode number. */
    } di_u;
    u_int64_t       di_size;        /*   8: File byte count. */
    int32_t         di_atime;       /*  16: Last access time. */
    int32_t         di_atimensec;   /*  20: Last access time. */
    int32_t         di_mtime;       /*  24: Last modified time. */
    int32_t         di_mtimensec;   /*  28: Last modified time. */
    int32_t         di_ctime;       /*  32: Last inode change time. */
    int32_t         di_ctimensec;   /*  36: Last inode change time. */
    ufs1_daddr_t    di_db[NDADDR];  /*  40: Direct disk blocks. */
    ufs1_daddr_t    di_ib[NIADDR];  /*  88: Indirect disk blocks. */
    u_int32_t       di_flags;       /* 100: Status flags (chflags). */
    int32_t         di_blocks;      /* 104: Blocks actually held. */
    int32_t         di_gen;         /* 108: Generation number. */
    u_int32_t       di_uid;         /* 112: File owner. */
    u_int32_t       di_gid;         /* 116: File group. */
    int32_t         di_spare[2];    /* 120: Reserved; currently unused */
};
```

inodos en openBSD

```
struct ufs2_dinode {
    u_int16_t       di_mode;        /*   0: IFMT, permissions; see below. */
    int16_t         di_nlink;       /*   2: File link count. */
    u_int32_t       di_uid;         /*   4: File owner. */
    u_int32_t       di_gid;         /*   8: File group. */
    u_int32_t       di_blksize;     /*  12: Inode blocksize. */
    u_int64_t       di_size;        /*  16: File byte count. */
    u_int64_t       di_blocks;      /*  24: Bytes actually held. */
    ufs_time_t      di_atime;       /*  32: Last access time. */
    ufs_time_t      di_mtime;       /*  40: Last modified time. */
    ufs_time_t      di_ctime;       /*  48: Last inode change time. */
    ufs_time_t      di_birthtime;   /*  56: Inode creation time. */
    int32_t         di_mtimensec;   /*  64: Last modified time. */
    int32_t         di_atimensec;   /*  68: Last access time. */
    int32_t         di_ctimensec;   /*  72: Last inode change time. */
    int32_t         di_birthnsec;   /*  76: Inode creation time. */
    int32_t         di_gen;         /*  80: Generation number. */
    u_int32_t       di_kernflags;   /*  84: Kernel flags. */
    u_int32_t       di_flags;       /*  88: Status flags (chflags). */
    int32_t         di_extsize;     /*  92: External attributes block. */
    ufs2_daddr_t    di_extb[NXADDR];/*  96: External attributes block. */
    ufs2_daddr_t    di_db[NDADDR];  /* 112: Direct disk blocks. */
    ufs2_daddr_t    di_ib[NIADDR];  /* 208: Indirect disk blocks. */
    int64_t         di_spare[3];    /* 232: Reserved; currently unused */
};
```

inodo en ext2

```
struct ext2fs_dinode {
    u_int16_t       e2di_mode;      /*   0: IFMT, permissions; see below. */
    u_int16_t       e2di_uid_low;   /*   2: Owner UID, lowest bits */
    u_int32_t       e2di_size;      /*      4: Size (in bytes) */
    u_int32_t       e2di_atime;     /*      8: Access time */
    u_int32_t       e2di_ctime;     /*     12: Create time */
    u_int32_t       e2di_mtime;     /*     16: Modification time */
    u_int32_t       e2di_dtime;     /*     20: Deletion time */
    u_int16_t       e2di_gid_low;   /*  24: Owner GID, lowest bits */
    u_int16_t       e2di_nlink;     /*  26: File link count */
    u_int32_t       e2di_nblock;    /*  28: Blocks count */
    u_int32_t       e2di_flags;     /*  32: Status flags (chflags) */
    u_int32_t       e2di_linux_reserved1; /* 36 */
    u_int32_t       e2di_blocks[NDADDR+NIADDR]; /* 40: disk blocks */
    u_int32_t       e2di_gen;       /* 100: generation number */
    u_int32_t       e2di_facl;      /* 104: file ACL (not implemented) */
    u_int32_t       e2di_dacl;      /* 108: dir ACL (not implemented) */
    u_int32_t       e2di_faddr;     /* 112: fragment address */
    u_int8_t        e2di_nfrag;     /* 116: fragment number */
    u_int8_t        e2di_fsize;     /* 117: fragment size */
    u_int16_t       e2di_linux_reserved2; /* 118 */
    u_int16_t       e2di_uid_high;  /* 120: 16 highest bits of uid */
    u_int16_t       e2di_gid_high;  /* 122: 16 highest bits of gid */
    u_int32_t       e2di_linux_reserved3; /* 124 */
};
```

Punto de vista del kernel

- un directorio es un fichero normal. Sus contenidos son las *entradas de directorio*, cada una de ellas contiene información de uno de los ficheros en dicho directorio. (basicamente el nombre y el número de inodo)
- cada fichero un único *inodo* pero varios nombres (enlaces)
  - enlace real a un fichero: entrada de directorio que se refiere al mismo inodo
  - enlace simbólico a un fichero: fichero especial que contiene el path al cual es el enlace
- varios tipos de fichero
  - fichero normal
  - directorio
  - dispositivo (bloque o carácter)
  - enlace simbólico
  - fifo
  - socket

constantes en ufs/ufs/dinode.h

```
#define NDADDR  12                      /* Direct addresses in inode. */
#define NIADDR  3                       /* Indirect addresses in inode. */

#define MAXSYMLINKLEN_UFS1      ((NDADDR + NIADDR) * sizeof(ufs1_daddr_t))
#define MAXSYMLINKLEN_UFS2      ((NDADDR + NIADDR) * sizeof(ufs2_daddr_t))

/* File permissions. */
#define IEXEC           0000100         /* Executable. */
#define IWRITE          0000200         /* Writeable. */
#define IREAD           0000400         /* Readable. */
#define ISVTX           0001000         /* Sticky bit. */
#define ISGID           0002000         /* Set-gid. */
#define ISUID           0004000         /* Set-uid. */

/* File types. */
#define IFMT            0170000         /* Mask of file type. */
#define IFIFO           0010000         /* Named pipe (fifo). */
#define IFCHR           0020000         /* Character device. */
#define IFDIR           0040000         /* Directory file. */
#define IFBLK           0060000         /* Block device. */
#define IFREG           0100000         /* Regular file. */
#define IFLNK           0120000         /* Symbolic link. */
#define IFSOCK          0140000         /* UNIX domain socket. */
#define IFWHT           0160000         /* Whiteout. */

#endif /* _UFS_DINODE_H_ */
```

- una instalación puede tener una o varias unidades físicas
- cada unidad física puede tener uno o varios sistemas de ficheros (o unidades lógicas)
- cada sistema de ficheros: sucesión de bloques (grupos de sectores) de 512, 1024, 2048 ...bytes. En Unix System V R2 tiene la siguiente estructuta física

BOOT	SUPER BLOQUE	LISTA INODOS	AREA DE DATOS

- los distintos sistemas de ficheros se *montan* (llamada al sistema *mount*) sobre directorios dando lugar a un único árbol (grafo) de directorios en el sistema.

- el kernel trata sólo con dispositivos lógicos.
  - Cada fichero en el sistema queda perfectamente definido por un número de dispositivo lógico (sistema de ficheros) y número de inodo dentro de ese sistema de ficheros
  - cada bloque queda perfectamente definido por un número de dispositivo lógico (sistema de ficheros) y número de bloque dentro de ese sistema de ficheros
  - las estrategias de asignación y contabilidad se hacen en base a bloques lógicos
- la traducción de direcciones lógicas a direcciones físicas la hace el manejador de dispositivo (*device driver*)

- El uso de ficheros en el sistema está gobernado por tres tablas
  - **tabla de inodos en memoria(inode table)** Contiene los inodos en memoria de los ficheros que están en uso, junto con, entre otras cosas, un contador de referencias. Global del sistema, en el espacio de datos del kernel
  - **tabla ficheros abiertos(file table)** Una entrada por cada apertura de un fichero (varias aperturas del mismo fichero dan lugar a varias entradas). Contiene el modo de apertura (O_RDONLY, O_WRONLY . . .), *offset* en el fichero, contador de referencias y puntero al inodo en la tabla de inodos en memoria. Global del sistema, en el espacio de datos del kernel
  - **tabla de descriptores de fichero de usuario (user file descriptor table)** Cada apertura, o cada llamada *dup()* crean una entrada en esta tabla (asi como la llamada *fork()*). Contiene un referencia a la entrada correspondiente de la tabla ficheros abiertos. Una para cada proceso, en su *u_area*

ejemplo de tablas de ficheros

Si tenemos dos procesos

- P1

```
...
df1=open("F1",O_RDONLY);
df2=open("F1",O_RDONLY);
df3=open("F2",O_RDWR);
..
```

- P2

```
...
df1=open("F2",O_WRONLY);
df2=dup(df1);
df3=open("F2",O_RDWR|O_APPEND);
..
```

ejemplo de tablas de ficheros