laSalle ENG
Universitat Ramon Llull

Sistemes Operatius

Curs 2016-2017

laSalle ENG
Universitat Ramon Llull

CONTINGUTS

1. File Descriptors.
2. Signals en C.

1. File descriptors

File Descriptor: element bàsic per a crides al sistema d'entrada sortida.

- File descriptor 0: entrada (habitualment teclat).
- File descriptor 1: sortida (habitualment pantalla).
- File descriptor 2: error (habitualment pantalla).

Entrada sortida bàsica

- Ja no usarem gets(), scanf(), printf() etc.
- Usarem només read() i write()
- Exemples:

```
printf("Hola\n");  →  write(1,"Hola\n",5);
scanf("%s",cadena);  →  n=read(0,cadena,100);
```

- Interessant per sortida: sprintf()

1. File descriptors

Treball amb arxius → fitxers NO bin o txt, fitxer de bits

- Ja no usarem fopen(), fclose(), fread, fwrite(), fgets(), fscanf(), fprintf() o feof().
- Usarem open(), close(), read() i write()

- Exemple:

```
FILE *f;
f=fopen("fitxer.dat","rb");
if (f==NULL)...

int fd;
fd=open("fitxer.dat",O_RDONLY);
If (fd<0)...
:
close(fd);
```

abans

ara!

← fileDescriptor

permisos!

↳ ens retorna el fileDescriptor que li pertoca.

↳ No hem pogut obrir, ja que ens han assignat un fileDescriptor negatiu

si fem un read del fitxer i ens retorna 0 → final de fitxer

↳ és com fem el feof()

printf ("El valor de x = %d\n", x);

Vaja!

sprintf (cadena, "el valor x = %d\n", x); → en ha posat dins la variable cadena "el valor x = 34 \n"

write (1, cadena, strlen (cadena));

1. Signals

- **Signal:** mecanisme del nucli per comunicar events als processors.
- Cada signal té associat un nom i un número (**kill -l**).
- **Funcions que podem usar:**
  - signal(int sig, sig_t funció)
  - kill(int pid, int sig)
  - raise(int sig)

té 64 interrupcions

1. Signals

```
int main()
{
        int i;
        for (i = 1; i <= 64; i++)
        {
                signal(i, ksighandler);
        }
        alarm(30);
        while (1)
        {
                printf("Soc el pid %d i espero senyals\n", getpid());
                sleep(2);
        }
}
```

acabem de reprogramar les 64 interrupcions i els hi ha posat la interrupció Ksighandler

té a veure amb la interrupció del clock → diem que d'aquí a 30 seg llenci una interrupció; com que abans hem reprogramat, llançarem Ksighander; ← No esperem els 30 seg aquí sent ....

1. Signals

↓ Funció que hem posat a les interrupcions

→ sempre ha de rebre el num d'interrupció

```
void ksighandler(int signum)
{
        switch (signum)
        {
                case SIGINT:   ← cntrl C
                        kctrlc(signum);
                        break;
                case SIGALRM:  ← interrupció rellotge
                        ksigalrm(signum);
                        break;
                default:
                        kgeneric(signum);
                        break;
        }
        signal(signum, ksighandler);
}
```

1. Signals

laSalle ENG
Universitat Ramon Llull

```
void kctrlc(int signum)
{
	printf("He rebut un ctrl + c (%d).\n", signum);
}

void ksigalrm(int signum)
{
	printf("He rebut un sigalrm (%d).\n", signum);
}

void kgeneric(int signum)
{
	printf("He rebut un altre senyal (%d)\n", signum);
}
```