

Fundamentos de Software de Comunicaciones

Tema 2 Programación del Sistema Operativo (4ª parte)

Motivación

- Las funciones de E/S (`read`, `write`) tienen por defecto un comportamiento bloqueante
 - **`read()`**: cuando se lee de un canal (descriptor) abierto, pero sin datos
 - **`write()`**: cuando se escribe en canal (descriptor) que está lleno
- En el software de comunicaciones es un escenario típico
 - Un servidor que atiende a múltiples clientes
 - Cada cliente usa un canal de comunicaciones que se gestiona con un descriptor de fichero
- Posibles soluciones (con las herramientas vistas hasta ahora)
 - Crear un hijo para atender a cada canal de forma independiente
 - **Demultiplexar la E/S**: escuchar por varios canales a la vez

Multiplexación de E/S

- Se utiliza la llamada al sistema `select()`
- Esta llamada duerme un proceso hasta que algún elemento de un conjunto de descriptores de E/S (ficheros regulares, pipes, sockets, ...) está preparado para:
 - hacer lecturas por él
 - hacer escrituras en él
 - notificar una situación excepcional sobre él
- **No lee/escrive en él, sólo indica cuando está listo**
- **Programación reactiva**

La función select()

```
int select(int      maxfds_mas_uno,
           fd_set * rfd,
           fd_set * wfd,
           fd_set * efd,
           struct timeval * timeout);
```

■ Descripción:

- **maxfds**: el valor más alto de entre los descriptors a analizar más uno (+1)
- **rfd**: dirección del conjunto de descriptors que se chequean para ver si están preparados para ser **leídos**, incluido si están cerrados (en cuyo caso la lectura no bloquea y devuelve **0**)
- **wfd**: dirección del conjunto de descriptors que se chequean para ver su disponibilidad de **escritura**
- **efd**: dirección del conjunto de descriptors que se chequean para ver si surge alguna **condición excepcional**. Por ejemplo, los datos URGENTES en un socket
- **timeout**: dirección del tiempo de espera antes de acabar, en caso de no haber ningún descriptor preparado (si vale **NULL**) la espera es infinita hasta que algún descriptor esté listo
- **Devuelve**: el número de descriptors listos, 0 si ha saltado el temporizador o -1 si hubo error

■ Existe mucha información en el manual

- **man 2 select**: para información básica de funcionamiento
- **man 2 select_tut**: para un tutorial más extenso sobre su uso

La funcion `select()`: argumentos

- El primer argumento de **`select`** le indica cual es el mayor descriptor de los que se van a chequear (más uno). Hay varias funciones/constantes del S.O. que ayudan, por ejemplo:
 - `int maxfds_mas_uno = getdtablesize();`
 - Directamente devuelve el número del mayor descriptor de fichero que un proceso puede usar, más uno.
 - La constante **`FD_SETSIZE`**
- Utilizar **`getdtablesize()`** es más ineficiente que calcular el máximo descriptor+1, puesto que **`select()`** tiene que consultar más descriptors de los seguramente necesarios (mirar en toda la tabla de descriptors de archivos del proceso)
- **La mejor opción** es buscar el máximo de entre los que se chequean (cada vez que se añade un nuevo descriptor a un conjunto **`fd_set`**) y sumarle uno
 - Definiendo una función máximo entre dos valores enteros:
`maxfds = max(fd, maxfds);`

Conjuntos de descriptores de fichero

- Se representan mediante el tipo de datos **fd_set**
- Se suelen implementar como una máscara de bits: un buffer de tamaño fijo **FD_SETSIZE**
 - Por defecto, en nuestro sistema **FD_SETSIZE = 1024**

- Ejemplo

- Asumiendo todo a cero a partir de la posición 4 en el siguiente **fd_set**



- Equivale al conjunto: **{0,3}**
- Es **obligatorio** que los descriptores incluidos en los **fd_set** sean descriptores válidos
 - Ficheros, pipes, etc. **abiertos previamente**

La funcion select(): argumentos

- Existen macros que permiten gestionar los conjuntos de descriptors:
 - **FD_ZERO(fd_set * xfd)**
 - Pone a cero (resetea) el conjunto **xfd**
 - Es **obligatorio** su uso antes de añadir ningún descriptor
 - Usando nomenclatura de conjuntos: **xfd = {}**
 - **FD_SET(int fd, fd_set * xfd)**
 - Añade el descriptor **fd** al conjunto **xfd**
 - Usando nomenclatura de conjuntos: **xfd = xfd U {fd}**
 - **FD_CLR(int fd, fd_set * xfd)**
 - Elimina el descriptor **fd** del conjunto **xfd**
 - Usando nomenclatura de conjuntos: **xfd = xfd \ {fd}**
 - **FD_ISSET(int fd, fd_set * xfd)**
 - Comprueba si el descriptor **fd** pertenece al conjunto **xfd** (devuelve un **valor distinto de 0** si está en el conjunto, 0 si no lo está)

La funcion select(): argumentos

- Es importante saber que select() **modifica los conjuntos que se le pasan como argumentos**
 - Cada vez que se llama a **select()** **NO** interesa hacerlo con los conjuntos originales, por lo que se deben hacer copias de los mismos si se vuelve a **select()** en un bucle
 - Las copias modificadas de conjuntos de descriptors devueltas por **select()** son las que se chequean con la macro **FD_ISSET**
- El argumento ***timeout*** indica el tiempo de espera en **select()**
 - Si vale **NULL**, la espera es infinita hasta que haya descriptors activos
 - Si no, hay que tener cuidado porque algunos sistemas operativos (como Linux) también modifican su valor a la salida
 - Lo más seguro es hacer una **copia** del timeout original y pasarla al **select** (como se hace con los conjuntos de descriptors)

La funcion `select()`: uso típico

1. Crear de los **fd_set** con los descriptors que se quiere comprobar que están listos para E/S
2. Copiar el conjunto original en un conjunto auxiliar con el que llamar a **select()**
3. Llamar a **select()**, que se duerme hasta que uno o más de esos descriptors están listos para E/S
4. Cuando se desbloquea, se ha de comprobar cuál de los descriptors está listo, consultando **TODOS** los descriptors del conjunto auxiliar modificado por **select()**
5. Una vez identificado el descriptor, entonces se puede realizar la lectura/escritura sin bloquear al proceso
OJO: `select()` no lee/escrive, sólo indica cuándo el descriptor está listo
6. Evaluar si es necesario actualizar el conjunto original de descriptors (por la detección de cierre de algún canal)
7. Ir al Paso 2

Ejemplo con select()

```
/* Parte de un programa con varias fifos simultaneas de las que se pueden leer datos
 */
int max_fd(int actual, int nuevo){ /*devuelve el mayor entre dos enteros*/
    if(actual >= nuevo)
        return actual;
    return nuevo;
}
void copia_fdset(fd_set *dst, fd_set *origen, int maxfd_mas_uno){
    FD_ZERO(dst);
    for(int i=0;i<maxfd_mas_uno;i++){
        if(FD_ISSET(i,origen))
            FD_SET(i,dst);
    }
}
int main(){
    fd_set rfd, active_rfd;
    /*...supongamos que inicialmente tenemos dos fifos ya abiertas: f1 y f2 */
    FD_ZERO(&rfd);
    FD_SET(f1, &rfd);
    FD_SET(f2, &rfd);
    while(1){
        /*cada vez que se llama a select se crea una copia del conjunto original*/
        copia_fdset(&active_rfd, &rfd, max_fd(f1,f2)+1);
        /*select comprueba si los descriptors están disponibles para lectura. Como no se
        ha establecido un timeout, se bloquea hasta que al menos uno esté disponible */
        int rc = select(max_fd(f1,f2)+1, &active_rfd, NULL, NULL, NULL);
    }
}
```

Ejemplo con select()

```
/* chequeo de error si rc < 0 ... */
if(rc < 0){
    perror("select");
    close(f1); close(f2);
    exit(1);
}
const int MAXBUFSIZE = 2048;
char buffer[MAXBUFSIZE];
/* si la fifo f1 tiene datos disponibles para leer ... */
if (FD_ISSET(f1, &active_rfd)) {
    read(f1, buffer, MAXBUFSIZE);
    /* ... código asociado ...*/
}
/* si la fifo f2 tiene datos disponibles para leer ... */
if (FD_ISSET(f2, &active_rfd)) {
    read(f2, buffer, MAXBUFSIZE);
    /* ... código asociado ...*/
}
} /* fin del while(1) */
return 0;
}
```

Ejemplo con select()

- Usaremos notación de conjuntos para representar los conjuntos de descriptores → **{f1,f2}** indica que los descriptores **f1** y **f2** están en el conjunto

rfd = {} //vacío

```
47 // Estructuras de datos para el select
48 fd_set rfd_a, rfd;
49
50 FD_ZERO(&rfd);
51 FD_SET(f1, &rfd);
52 FD_SET(f2, &rfd);
53
54 int maximo = f1 > f2 ? f1:f2;
55 struct timeval t, t_a;
56 t.tv_sec = 2; t.tv_usec = 750000;
57
58 while (1) {
59     copia_fdset(&rfd_a, &rfd, maximo+1);
60     t_a = t;
61     if ( (r = select(maximo+1, &rfd_a, NULL, NULL, &t_a)) < 0) {
62         perror("select");
63         close(f2); close(f1); exit(-1);
64     }
65
66     if (r == 0) {
67         /* Se realiza la acción especificada. Ojo: read bloquearía */
68     }
69
70     if (FD_ISSET(f1, &rfd_a)) {
71         leidos = read(f1, b, T);
72         /* proceso de la información recibida */
73     }
74
75     if (FD_ISSET(f2, &rfd_a)) {
76         leidos = read(f2, b, T);
77         /* proceso de la información recibida */
78     }
79 }
```

Ejemplo con select()

- Usaremos notación de conjuntos para representar los conjuntos de descriptores → **{f1,f2}** indica que los descriptores **f1** y **f2** están en el conjunto

rfd = { **f1**, **f2** }

```
47 // Estructuras de datos para el select
48 fd_set rfd_a, rfd;
49
50 FD_ZERO(&rfd);
51 FD_SET(f1, &rfd);
52 FD_SET(f2, &rfd);
53
54 int maximo = f1 > f2 ? f1:f2;
55 struct timeval t, t_a;
56 t.tv_sec = 2; t.tv_usec = 750000;
57
58 while (1) {
59     copia_fdset(&rfd_a, &rfd, maximo+1);
60     t_a = t;
61     if ( (r = select(maximo+1, &rfd_a, NULL, NULL, &t_a)) < 0 ) {
62         perror("select");
63         close(f2); close(f1); exit(-1);
64     }
65
66     if (r == 0) {
67         /* Se realiza la acción especificada. Ojo: read bloquearía */
68     }
69
70     if (FD_ISSET(f1, &rfd_a)) {
71         leidos = read(f1, b, T);
72         /* proceso de la información recibida */
73     }
74
75     if (FD_ISSET(f2, &rfd_a)) {
76         leidos = read(f2, b, T);
77         /* proceso de la información recibida */
78     }
79 }
```

Ejemplo con select()

- Usaremos notación de conjuntos para representar los conjuntos de descriptores → **{f1,f2}** indica que los descriptores **f1** y **f2** están en el conjunto
- Sea **f1 = 3** y **f2 = 4**

Primera iteración

rfd = {f1, f2}
maximo = 4
t = 2.75 s

```
47 // Estructuras de datos para el select
48 fd_set rfd_a, rfd;
49
50 FD_ZERO(&rfd);
51 FD_SET(f1, &rfd);
52 FD_SET(f2, &rfd);
53
54 int maximo = f1 > f2 ? f1:f2;
55 struct timeval t, t_a;
56 t.tv_sec = 2; t.tv_usec = 750000;
57
58 while (1) {
59     copia_fdset(&rfd_a, &rfd, maximo+1);
60     t_a = t;
61     if ( (r = select(maximo+1, &rfd_a, NULL, NULL, &t_a)) < 0) {
62         perror("select");
63         close(f2); close(f1); exit(-1);
64     }
65
66     if (r == 0) {
67         /* Se realiza la acción especificada. Ojo: read bloquearía */
68     }
69
70     if (FD_ISSET(f1, &rfd_a)) {
71         leidos = read(f1, b, T);
72         /* proceso de la información recibida */
73     }
74
75     if (FD_ISSET(f2, &rfd_a)) {
76         leidos = read(f2, b, T);
77         /* proceso de la información recibida */
78     }
79 }
```

Ejemplo con select()

- Usaremos notación de conjuntos para representar los conjuntos de descriptors → **{f1,f2}** indica que los descriptors **f1** y **f2** están en el conjunto
- Sea **f1 = 3** y **f2 = 4** → **maximo = 4**

Primera iteración

- Se crea una copia del conjunto **rfd** y del **timeval**

rfd = {f1, f2}
rfd_a = {f1, f2}
t_a = 2.75 s

```
47 // Estructuras de datos para el select
48 fd_set rfd_a, rfd;
49
50 FD_ZERO(&rfd);
51 FD_SET(f1, &rfd);
52 FD_SET(f2, &rfd);
53
54 int maximo = f1 > f2 ? f1:f2;
55 struct timeval t, t_a;
56 t.tv_sec = 2; t.tv_usec = 750000;
57
58 while (1) {
59     copia_fdset(&rfd_a, &rfd, maximo+1);
60     t_a = t;
61     if ( (r = select(maximo+1, &rfd_a, NULL, NULL, &t_a)) < 0 ) {
62         perror("select");
63         close(f2); close(f1); exit(-1);
64     }
65
66     if (r == 0) {
67         /* Se realiza la acción especificada. Ojo: read bloquearía */
68     }
69
70     if (FD_ISSET(f1, &rfd_a)) {
71         leidos = read(f1, b, T);
72         /* proceso de la información recibida */
73     }
74
75     if (FD_ISSET(f2, &rfd_a)) {
76         leidos = read(f2, b, T);
77         /* proceso de la información recibida */
78     }
79 }
```

Ejemplo con select()

- Usaremos notación de conjuntos para representar los conjuntos de descriptors → **{f1,f2}** indica que los descriptors **f1** y **f2** están en el conjunto
- Sea **f1 = 3** y **f2 = 4** → **maximo = 4**

Primera iteración

- Se crea una copia del conjunto **rfd** y del **timeval**
- **f1** está listo para lectura en **1.25 s**, **select** actualiza **rfd_a** y **t_a**

rfd	=	{f1, f2}
rfd_a	=	{ f1 }
t_a	=	1.5 s
r	=	1

```
47 // Estructuras de datos para el select
48 fd_set rfd_a, rfd;
49
50 FD_ZERO(&rfd);
51 FD_SET(f1, &rfd);
52 FD_SET(f2, &rfd);
53
54 int maximo = f1 > f2 ? f1:f2;
55 struct timeval t, t_a;
56 t.tv_sec = 2; t.tv_usec = 750000;
57
58 while (1) {
59     copia_fdset(&rfd_a, &rfd, maximo+1);
60     t_a = t;
61     if ( (r = select(maximo+1, &rfd_a, NULL, NULL, &t_a)) < 0) {
62         perror("select");
63         close(f2); close(f1); exit(-1);
64     }
65
66     if (r == 0) {
67         /* Se realiza la acción especificada. Ojo: read bloquearía */
68     }
69
70     if (FD_ISSET(f1, &rfd_a)) {
71         leidos = read(f1, b, T);
72         /* proceso de la información recibida */
73     }
74
75     if (FD_ISSET(f2, &rfd_a)) {
76         leidos = read(f2, b, T);
77         /* proceso de la información recibida */
78     }
79 }
```


Ejemplo con select()

- Usaremos notación de conjuntos para representar los conjuntos de descriptors → **{f1,f2}** indica que los descriptors **f1** y **f2** están en el conjunto
- Sea **f1 = 3** y **f2 = 4** → **maximo = 4**

Primera iteración

- Se crea una copia del conjunto **rfd** y del **timeval**
- **f1** está listo para lectura en **1.25 s**, **select** actualiza **rfd_a** y **t_a**
- El timeout no ha expirado (**t_a = 1.5 s**) → esta condición no es cierta

rfd	=	{f1, f2}
rfd_a	=	{ f1 }
t_a	=	1.5 s
r	=	1

```
47 // Estructuras de datos para el select
48 fd_set rfd_a, rfd;
49
50 FD_ZERO(&rfd);
51 FD_SET(f1, &rfd);
52 FD_SET(f2, &rfd);
53
54 int maximo = f1 > f2 ? f1:f2;
55 struct timeval t, t_a;
56 t.tv_sec = 2; t.tv_usec = 750000;
57
58 while (1) {
59     copia_fdset(&rfd_a, &rfd, maximo+1);
60     t_a = t;
61     if ( (r = select(maximo+1, &rfd_a, NULL, NULL, &t_a)) < 0 ) {
62         perror("select");
63         close(f2); close(f1); exit(-1);
64     }
65
66     if (r == 0) {
67         /* Se realiza la acción especificada. Ojo: read bloquearía */
68     }
69
70     if (FD_ISSET(f1, &rfd_a)) {
71         leidos = read(f1, b, T);
72         /* proceso de la información recibida */
73     }
74
75     if (FD_ISSET(f2, &rfd_a)) {
76         leidos = read(f2, b, T);
77         /* proceso de la información recibida */
78     }
79 }
```

Ejemplo con select()

- Usaremos notación de conjuntos para representar los conjuntos de descriptors → **{f1,f2}** indica que los descriptors **f1** y **f2** están en el conjunto
- Sea **f1 = 3** y **f2 = 4** → **maximo = 4**

Primera iteración

- Se crea una copia del conjunto **rfd** y del **timeval**
- **f1** está listo para lectura en **1.25 s**, **select** actualiza **rfd_a** y **t_a**
- **t_a = 1.5 s**, **r = 1** → esta condición no es cierta
- El primer **FD_ISSET** devuelve un valor verdadero → se lee de **f1** sin bloquear el **read()**

```
rfd    = {f1, f2}
rfd_a  = {f1}
t_a    = 1.5 s
r      = 1
```

```
47 // Estructuras de datos para el select
48 fd_set rfd_a, rfd;
49
50 FD_ZERO(&rfd);
51 FD_SET(f1, &rfd);
52 FD_SET(f2, &rfd);
53
54 int maximo = f1 > f2 ? f1:f2;
55 struct timeval t, t_a;
56 t.tv_sec = 2; t.tv_usec = 750000;
57
58 while (1) {
59     copia_fdset(&rfd_a, &rfd, maximo+1);
60     t_a = t;
61     if ( (r = select(maximo+1, &rfd_a, NULL, NULL, &t_a)) < 0 ) {
62         perror("select");
63         close(f2); close(f1); exit(-1);
64     }
65
66     if (r == 0) {
67         /* Se realiza la acción especificada. Ojo: read bloquearía */
68     }
69
70     if (FD_ISSET(f1, &rfd_a)) {
71         leidos = read(f1, b, T);
72         /* proceso de la información recibida */
73     }
74
75     if (FD_ISSET(f2, &rfd_a)) {
76         leidos = read(f2, b, T);
77         /* proceso de la información recibida */
78     }
79 }
```

Ejemplo con select()

- Usaremos notación de conjuntos para representar los conjuntos de descriptors → **{f1,f2}** indica que los descriptors **f1** y **f2** están en el conjunto
- Sea **f1 = 3** y **f2 = 4** → **maximo = 4**

Primera iteración

- Se crea una copia del conjunto **rfd** y del **timeval**
- **f1** está listo para lectura en **1.25 s**, **select** actualiza **rfd_a** y **t_a**
- **t_a = 1.5 s**, **r = 1** → esta condición no es cierta
- El primer **FD_ISSET** devuelve un valor verdadero → se lee de **f1** sin bloquear el **read()**
- El segundo devuelve un valor falso (**f2** no está en el **rfd_a**) y no se emite una lectura, puesto que bloquearía al proceso

```
rfd    = { f1, f2 }  
rfd_a  = { f1 }  
t_a    = 1.5 s  
r      = 1
```

```
47 // Estructuras de datos para el select  
48 fd_set rfd_a, rfd;  
49  
50 FD_ZERO(&rfd);  
51 FD_SET(f1, &rfd);  
52 FD_SET(f2, &rfd);  
53  
54 int maximo = f1 > f2 ? f1:f2;  
55 struct timeval t, t_a;  
56 t.tv_sec = 2; t.tv_usec = 750000;  
57  
58 while (1) {  
59     copia_fdset(&rfd_a, &rfd, maximo+1);  
60     t_a = t;  
61     if ( (r = select(maximo+1, &rfd_a, NULL, NULL, &t_a)) < 0 ) {  
62         perror("select");  
63         close(f2); close(f1); exit(-1);  
64     }  
65  
66     if (r == 0) {  
67         /* Se realiza la acción especificada. Ojo: read bloquearía */  
68     }  
69  
70     if (FD_ISSET(f1, &rfd_a)) {  
71         leidos = read(f1, b, T);  
72         /* proceso de la información recibida */  
73     }  
74  
75     if (FD_ISSET(f2, &rfd_a)) {  
76         leidos = read(f2, b, T);  
77         /* proceso de la información recibida */  
78     }  
79 }
```

Ejemplo con select()

- Usaremos notación de conjuntos para representar los conjuntos de descriptors → **{f1,f2}** indica que los descriptors **f1** y **f2** están en el conjunto
- Sea **f1 = 3** y **f2 = 4** → **maximo = 4**

Segunda iteración

- Se copia de nuevo el conjunto **rfd** y el **timeval**

rfd = {f1, f2}
rfd_a = {f1, f2}
t_a = 2.75 s

```
47 // Estructuras de datos para el select
48 fd_set rfd_a, rfd;
49
50 FD_ZERO(&rfd);
51 FD_SET(f1, &rfd);
52 FD_SET(f2, &rfd);
53
54 int maximo = f1 > f2 ? f1:f2;
55 struct timeval t, t_a;
56 t.tv_sec = 2; t.tv_usec = 750000;
57
58 while (1) {
59     copia_fdset(&rfd_a, &rfd, maximo+1);
60     t_a = t;
61     if ( (r = select(maximo+1, &rfd_a, NULL, NULL, &t_a)) < 0) {
62         perror("select");
63         close(f2); close(f1); exit(-1);
64     }
65
66     if (r == 0) {
67         /* Se realiza la acción especificada. Ojo: read bloquearía */
68     }
69
70     if (FD_ISSET(f1, &rfd_a)) {
71         leidos = read(f1, b, T);
72         /* proceso de la información recibida */
73     }
74
75     if (FD_ISSET(f2, &rfd_a)) {
76         leidos = read(f2, b, T);
77         /* proceso de la información recibida */
78     }
79 }
```

Ejemplo con select()

- Usaremos notación de conjuntos para representar los conjuntos de descriptors → **{f1,f2}** indica que los descriptors **f1** y **f2** están en el conjunto
- Sea **f1 = 3** y **f2 = 4** → **maximo = 4**

Segunda iteración

- Se copia de nuevo el conjunto **rfd** y el **timeval**
- Ahora **f1** y **f2** están listos para lectura en **2 s**, **select** actualiza **rfd_a**, **t_a** y **r**

```
rfd      = {f1, f2}
rfd_a    = {f1, f2}
t_a      = 0.75 s
r        = 2
```

```
47 // Estructuras de datos para el select
48 fd_set rfd_a, rfd;
49
50 FD_ZERO(&rfd);
51 FD_SET(f1, &rfd);
52 FD_SET(f2, &rfd);
53
54 int maximo = f1 > f2 ? f1:f2;
55 struct timeval t, t_a;
56 t.tv_sec = 2; t.tv_usec = 750000;
57
58 while (1) {
59     copia_fdset(&rfd_a, &rfd, maximo+1);
60     t_a = t;
61     if ( (r = select(maximo+1, &rfd_a, NULL, NULL, &t_a)) < 0) {
62         perror("select");
63         close(f2); close(f1); exit(-1);
64     }
65
66     if (r == 0) {
67         /* Se realiza la acción especificada. Ojo: read bloquearía */
68     }
69
70     if (FD_ISSET(f1, &rfd_a)) {
71         leidos = read(f1, b, T);
72         /* proceso de la información recibida */
73     }
74
75     if (FD_ISSET(f2, &rfd_a)) {
76         leidos = read(f2, b, T);
77         /* proceso de la información recibida */
78     }
79 }
```

Ejemplo con select()

- Usaremos notación de conjuntos para representar los conjuntos de descriptors → **{f1,f2}** indica que los descriptors **f1** y **f2** están en el conjunto
- Sea **f1 = 3** y **f2 = 4** → **maximo = 4**

Segunda iteración

- Se copia de nuevo el conjunto **rfd** y el **timeval**
- Ahora **f1** y **f2** están listos para lectura en **2 s**, **select** actualiza **rfd_a**, **t_a** y **r**
- **FD_ISSET** indica que se puede leer tanto de **f1**

rfd	=	{f1, f2}
rfd_a	=	{ f1 , f2 }
t_a	=	0.75 s
r	=	2

```
47 // Estructuras de datos para el select
48 fd_set rfd_a, rfd;
49
50 FD_ZERO(&rfd);
51 FD_SET(f1, &rfd);
52 FD_SET(f2, &rfd);
53
54 int maximo = f1 > f2 ? f1:f2;
55 struct timeval t, t_a;
56 t.tv_sec = 2; t.tv_usec = 750000;
57
58 while (1) {
59     copia_fdset(&rfd_a, &rfd, maximo+1);
60     t_a = t;
61     if ( (r = select(maximo+1, &rfd_a, NULL, NULL, &t_a)) < 0) {
62         perror("select");
63         close(f2); close(f1); exit(-1);
64     }
65
66     if (r == 0) {
67         /* Se realiza la acción especificada. Ojo: read bloquearía */
68     }
69
70     if (FD_ISSET(f1, &rfd_a)) {
71         leidos = read(f1, b, T);
72         /* proceso de la información recibida */
73     }
74
75     if (FD_ISSET(f2, &rfd_a)) {
76         leidos = read(f2, b, T);
77         /* proceso de la información recibida */
78     }
79 }
```

Ejemplo con select()

- Usaremos notación de conjuntos para representar los conjuntos de descriptors → **{f1,f2}** indica que los descriptors **f1** y **f2** están en el conjunto
- Sea **f1 = 3** y **f2 = 4** → **maximo = 4**

Segunda iteración

- Se copia de nuevo el conjunto **rfd** y el **timeval**
- Ahora **f1** y **f2** están listos para lectura en **2 s**, **select** actualiza **rfd_a**, **t_a** y **r**
- **FD_ISSET** indica que se puede leer tanto de **f1**
- ... como de **f2**
- En ningún caso, **read()** bloquea al proceso

rfd = {f1, f2}
rfd_a = {f1, f2}
t_a = 0.75 s
r = 2

```
47 // Estructuras de datos para el select
48 fd_set rfd_a, rfd;
49
50 FD_ZERO(&rfd);
51 FD_SET(f1, &rfd);
52 FD_SET(f2, &rfd);
53
54 int maximo = f1 > f2 ? f1:f2;
55 struct timeval t, t_a;
56 t.tv_sec = 2; t.tv_usec = 750000;
57
58 while (1) {
59     copia_fdset(&rfd_a, &rfd, maximo+1);
60     t_a = t;
61     if ( (r = select(maximo+1, &rfd_a, NULL, NULL, &t_a)) < 0 ) {
62         perror("select");
63         close(f2); close(f1); exit(-1);
64     }
65
66     if (r == 0) {
67         /* Se realiza la acción especificada. Ojo: read bloquearía */
68     }
69
70     if (FD_ISSET(f1, &rfd_a)) {
71         leidos = read(f1, b, T);
72         /* proceso de la información recibida */
73     }
74
75     if (FD_ISSET(f2, &rfd_a)) {
76         leidos = read(f2, b, T);
77         /* proceso de la información recibida */
78     }
79 }
```

Ejemplo con select()

- Usaremos notación de conjuntos para representar los conjuntos de descriptors → **{f1,f2}** indica que los descriptors **f1** y **f2** están en el conjunto
- Sea **f1 = 3** y **f2 = 4** → **maximo = 4**

Tercera iteración

- Se copia de nuevo el conjunto **rfd** y el **timeval**

rfd = {f1, f2}
rfd_a = {f1, f2}
t_a = 2.75 s

```
47 // Estructuras de datos para el select
48 fd_set rfd_a, rfd;
49
50 FD_ZERO(&rfd);
51 FD_SET(f1, &rfd);
52 FD_SET(f2, &rfd);
53
54 int maximo = f1 > f2 ? f1:f2;
55 struct timeval t, t_a;
56 t.tv_sec = 2; t.tv_usec = 750000;
57
58 while (1) {
59     copia_fdset(&rfd_a, &rfd, maximo+1);
60     t_a = t;
61     if ( (r = select(maximo+1, &rfd_a, NULL, NULL, &t_a)) < 0) {
62         perror("select");
63         close(f2); close(f1); exit(-1);
64     }
65
66     if (r == 0) {
67         /* Se realiza la acción especificada. Ojo: read bloquearía */
68     }
69
70     if (FD_ISSET(f1, &rfd_a)) {
71         leidos = read(f1, b, T);
72         /* proceso de la información recibida */
73     }
74
75     if (FD_ISSET(f2, &rfd_a)) {
76         leidos = read(f2, b, T);
77         /* proceso de la información recibida */
78     }
79 }
```


Ejemplo con select()

- Usaremos notación de conjuntos para representar los conjuntos de descriptors → **{f1,f2}** indica que los descriptors **f1** y **f2** están en el conjunto
- Sea **f1 = 3** y **f2 = 4** → **maximo = 4**

Tercera iteración

- Se copia de nuevo el conjunto **rfd** y el **timeval**
- Ahora pasan **2.75 s** sin que ninguna fifo esté lista para lectura
 - El conjunto está **vacío**
 - El timeval está a **cero**
 - Y **select** devuelve **cero**, indicando que no hay descriptor listo para lectura

rfd	=	{f1, f2}
rfd_a	=	{} //vacío
t_a	=	0 //expirado
r	=	0 //retorno

```
47 // Estructuras de datos para el select
48 fd_set rfd_a, rfd;
49
50 FD_ZERO(&rfd);
51 FD_SET(f1, &rfd);
52 FD_SET(f2, &rfd);
53
54 int maximo = f1 > f2 ? f1:f2;
55 struct timeval t, t_a;
56 t.tv_sec = 2; t.tv_usec = 750000;
57
58 while (1) {
59     copia_fdset(&rfd_a, &rfd, maximo+1);
60     t_a = t;
61     if ( (r = select(maximo+1, &rfd_a, NULL, NULL, &t_a)) < 0) {
62         perror("select");
63         close(f2); close(f1); exit(-1);
64     }
65
66     if (r == 0) {
67         /* Se realiza la acción especificada. Ojo: read bloquearía */
68     }
69
70     if (FD_ISSET(f1, &rfd_a)) {
71         leidos = read(f1, b, T);
72         /* proceso de la información recibida */
73     }
74
75     if (FD_ISSET(f2, &rfd_a)) {
76         leidos = read(f2, b, T);
77         /* proceso de la información recibida */
78     }
79 }
```

Ejemplo con select()

- Usaremos notación de conjuntos para representar los conjuntos de descriptors → **{f1,f2}** indica que los descriptors **f1** y **f2** están en el conjunto
- Sea **f1 = 3** y **f2 = 4** → **maximo = 4**

Tercera iteración

- Se copia de nuevo el conjunto **rfd** y el **timeval**
- Ahora pasan **2.75 s** sin que ninguna fifo esté lista para lectura
 - El conjunto está **vacío**
 - El timeval está a **cero**
 - Y **select** devuelve **cero**, indicando que no hay descriptor listo para lectura
- Ahora esta condición es **cierta**

rfd	=	{f1, f2}
rfd_a	=	{ } //vacío
t_a	=	0 //expirado
r	=	0 //retorno

```
47 // Estructuras de datos para el select
48 fd_set rfd_a, rfd;
49
50 FD_ZERO(&rfd);
51 FD_SET(f1, &rfd);
52 FD_SET(f2, &rfd);
53
54 int maximo = f1 > f2 ? f1:f2;
55 struct timeval t, t_a;
56 t.tv_sec = 2; t.tv_usec = 750000;
57
58 while (1) {
59     copia_fdset(&rfd_a, &rfd, maximo+1);
60     t_a = t;
61     if ( (r = select(maximo+1, &rfd_a, NULL, NULL, &t_a)) < 0 ) {
62         perror("select");
63         close(f2); close(f1); exit(-1);
64     }
65
66     if (r == 0) {
67         /* Se realiza la acción especificada. Ojo: read bloquearía */
68     }
69
70     if (FD_ISSET(f1, &rfd_a)) {
71         leidos = read(f1, b, T);
72         /* proceso de la información recibida */
73     }
74
75     if (FD_ISSET(f2, &rfd_a)) {
76         leidos = read(f2, b, T);
77         /* proceso de la información recibida */
78     }
79 }
```

Ejemplo con select()

- Usaremos notación de conjuntos para representar los conjuntos de descriptors → **{f1,f2}** indica que los descriptors **f1** y **f2** están en el conjunto
- Sea **f1 = 3** y **f2 = 4** → **maximo = 4**

Tercera iteración

- Se copia de nuevo el conjunto **rfd** y el **timeval**
- Ahora pasan **2.75 s** sin que ninguna fifo esté lista para lectura
 - El conjunto está **vacío**
 - El timeval está a **cero**
 - Y **select** devuelve **cero**, indicando que no hay descriptor listo para lectura
- Ahora esta condición es **cierta**
- Pero ninguno de los **FD_ISSET**, por lo que no se ejecuta ningún **read()** que bloquearía al proceso

```
47 // Estructuras de datos para el select
48 fd_set rfd_a, rfd;
49
50 FD_ZERO(&rfd);
51 FD_SET(f1, &rfd);
52 FD_SET(f2, &rfd);
53
54 int maximo = f1 > f2 ? f1:f2;
55 struct timeval t, t_a;
56 t.tv_sec = 2; t.tv_usec = 750000;
57
58 while (1) {
59     copia_fdset(&rfd_a, &rfd, maximo+1);
60     t_a = t;
61     if ( (r = select(maximo+1, &rfd_a, NULL, NULL, &t_a)) < 0) {
62         perror("select");
63         close(f2); close(f1); exit(-1);
64     }
65
66     if (r == 0) {
67         /* Se realiza la acción especificada. Ojo: read bloquearía */
68     }
69
70     if (FD_ISSET(f1, &rfd_a)) {
71         leidos = read(f1, b, T);
72         /* proceso de la información recibida */
73     }
74
75     if (FD_ISSET(f2, &rfd_a)) {
76         leidos = read(f2, b, T);
77         /* proceso de la información recibida */
78     }
79 }
```

Sobre select y el cierre de un descriptor

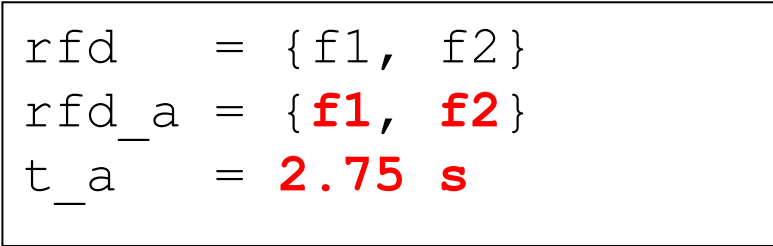
- En el ejemplo anterior, también se sale del bloqueo de **select()** cuando otro proceso cierre alguna de las fifos
 - Es decir, el otro proceso ya no va a hacer más operaciones de escritura en esa fifo
- Puesto que la operación **FD_ISSET** sobre el descriptor implicado en el **select()** ahora devolverá **1**, en el código se procederá a llamar a **read()** sobre dicho descriptor
- En este caso, **read()** tampoco será bloqueante y devolverá **cero**, lo que se interpretará como que el otro extremo de la fifo se ha cerrado y no se van a volver a recibir datos
 - Este es el momento de utilizar **FD_CLR** para quitar el descriptor del conjunto que debe monitorizar **select()** en la siguiente iteración del bucle **while**
- Este comportamiento es válido para pipes, fifos y sockets

Ejemplo con select(): cierre de un descriptor

- Continuando con el ejemplo anterior, ahora se especifica un poco más la lectura desde el descriptor en caso que se desbloquea
- El flujo de ejecución depende del valor devuelto por **read()**, que permite determinar cuándo actualizar el conjunto original de descriptors **rfd**

Cuarta iteración

- Se copia de nuevo el conjunto **rfd** y el **timeval**



```
rfd      = {f1, f2}
rfd_a    = {f1, f2}
t_a      = 2.75 s
```

```
58 while (1) {
59     copia_fdset(&rfd_a, &rfd, maximo+1);
60     t_a = t;
61     if ( (r = select(maximo+1, &rfd_a, NULL, NULL, &t_a)) < 0) {
62         perror("select"); close(f2); close(f1); exit(-1);
63     }
64
65     if (r == 0) {
66         /* Se realiza la acción especificada. Ojo: read bloquearía */
67     }
68
69     if (FD_ISSET(f1, &rfd_a)) {
70         leidos = read(f1, b, T);
71         if (leidos < 0) {
72             perror("read"); exit(-1);
73         } else if (leidos == 0) {
74             FD_CLR(f1, &rfd);
75         } else {
76             /* proceso de la información recibida */
77         }
78     }
79
80     if (FD_ISSET(f2, &rfd_a)) {
81         leidos = read(f2, b, T);
82         if (leidos < 0) {
83             perror("read"); exit(-1);
84         } else if (leidos == 0) {
85             FD_CLR(f2, &rfd);
86         } else {
87             /* proceso de la información recibida */
88         }
89     }
90 }
```

Ejemplo con select(): cierre de un descriptor

- Continuando con el ejemplo anterior, ahora se especifica un poco más la lectura desde el descriptor en caso que se desbloquee
- El flujo de ejecución depende del valor devuelto por **read()**, que permite determinar cuándo actualizar el conjunto original de descriptors **rfd**

Cuarta iteración

- Se copia de nuevo el conjunto **rfd** y el **timeval**
- Imaginemos que **select()** retorna porque **f2** está listo para lectura después de **1.75 segundos**

rfd	=	{f1, f2}
rfd_a	=	{f2}
t_a	=	1.00 s
r	=	1

```
58 while (1) {
59     copia_fdset(&rfd_a, &rfd, maximo+1);
60     t_a = t;
61     if ( (r = select(maximo+1, &rfd_a, NULL, NULL, &t_a)) < 0) {
62         perror("select"); close(f2); close(f1); exit(-1);
63     }
64
65     if (r == 0) {
66         /* Se realiza la acción especificada. Ojo: read bloquearía */
67     }
68
69     if (FD_ISSET(f1, &rfd_a)) {
70         leidos = read(f1, b, T);
71         if (leidos < 0) {
72             perror("read"); exit(-1);
73         } else if (leidos == 0) {
74             FD_CLR(f1, &rfd);
75         } else {
76             /* proceso de la información recibida */
77         }
78     }
79
80     if (FD_ISSET(f2, &rfd_a)) {
81         leidos = read(f2, b, T);
82         if (leidos < 0) {
83             perror("read"); exit(-1);
84         } else if (leidos == 0) {
85             FD_CLR(f2, &rfd);
86         } else {
87             /* proceso de la información recibida */
88         }
89     }
90 }
```

Ejemplo con select(): cierre de un descriptor

- Continuando con el ejemplo anterior, ahora se especifica un poco más la lectura desde el descriptor en caso que se desbloquea
- El flujo de ejecución depende del valor devuelto por **read()**, que permite determinar cuándo actualizar el conjunto original de descriptors **rfd**

Cuarta iteración

- Se copia de nuevo el conjunto **rfd** y el **timeval**
- Imaginemos que **select()** retorna porque **f2** está listo para lectura después de **1.75 segundos**
- La única condición verdadera es, por por tanto, la de la **línea 80**
- Imaginemos que la lectura siguiente **devuelve 0**, es decir, la tubería está **cerrada**

```
rfd      = {f1, f2}
rfd_a    = {f2}
t_a      = 1.00 s
leidos   = 0
```

```
58 while (1) {
59     copia_fdset(&rfd_a, &rfd, maximo+1);
60     t_a = t;
61     if ( (r = select(maximo+1, &rfd_a, NULL, NULL, &t_a)) < 0) {
62         perror("select"); close(f2); close(f1); exit(-1);
63     }
64
65     if (r == 0) {
66         /* Se realiza la acción especificada. Ojo: read bloquearía */
67     }
68
69     if (FD_ISSET(f1, &rfd_a)) {
70         leidos = read(f1, b, T);
71         if (leidos < 0) {
72             perror("read"); exit(-1);
73         } else if (leidos == 0) {
74             FD_CLR(f1, &rfd);
75         } else {
76             /* proceso de la información recibida */
77         }
78     }
79
80     if (FD_ISSET(f2, &rfd_a)) {
81         leidos = read(f2, b, T);
82         if (leidos < 0) {
83             perror("read"); exit(-1);
84         } else if (leidos == 0) {
85             FD_CLR(f1, &rfd);
86         } else {
87             /* proceso de la información recibida */
88         }
89     }
90 }
```

Ejemplo con select(): cierre de un descriptor

- Continuando con el ejemplo anterior, ahora se especifica un poco más la lectura desde el descriptor en caso que se desbloquee
- El flujo de ejecución depende del valor devuelto por **read()**, que permite determinar cuándo actualizar el conjunto original de descriptors **rfd**

Cuarta iteración

- Se copia de nuevo el conjunto **rfd** y el **timeval**
- Imaginemos que **select()** retorna porque **f2** está listo para lectura después de **1.75 segundos**
- La única condición verdadera es, por tanto, la de la **línea 80**
- Imaginemos que la lectura siguiente **devuelve 0**, es decir, la tubería está **cerrada**
- En ese caso, se puede sacar el descriptor del conjunto original → **No van a llegar más datos por ahí**

```
rfd      = {f1}
rfd_a    = {f2}
t_a      = 1.00 s
leidos   = 0
```

```
58 while (1) {
59     copia_fdset(&rfd_a, &rfd, maximo+1);
60     t_a = t;
61     if ( (r = select(maximo+1, &rfd_a, NULL, NULL, &t_a)) < 0) {
62         perror("select"); close(f2); close(f1); exit(-1);
63     }
64
65     if (r == 0) {
66         /* Se realiza la acción especificada. Ojo: read bloquearía */
67     }
68
69     if (FD_ISSET(f1, &rfd_a)) {
70         leidos = read(f1, b, T);
71         if (leidos < 0) {
72             perror("read"); exit(-1);
73         } else if (leidos == 0) {
74             FD_CLR(f1, &rfd);
75         } else {
76             /* proceso de la información recibida */
77         }
78     }
79
80     if (FD_ISSET(f2, &rfd_a)) {
81         leidos = read(f2, b, T);
82         if (leidos < 0) {
83             perror("read"); exit(-1);
84         } else if (leidos == 0) {
85             FD_CLR(f1, &rfd);
86         } else {
87             /* proceso de la información recibida */
88         }
89     }
90 }
```


Ejemplo con select(): cierre de un descriptor

- Continuando con el ejemplo anterior, ahora se especifica un poco más la lectura desde el descriptor en caso que se desbloquee
- El flujo de ejecución depende del valor devuelto por **read()**, que permite determinar cuándo actualizar el conjunto original de descriptors **rfd**

Quinta iteración

- Se copia de nuevo el conjunto **rfd** y el **timeval**: Ahora sólo hay un descriptor en **rfd**

```
rfd      = { f1 }  
rfd_a    = { f1 }  
t_a      = 2.75 s
```

- En ningún caso se puede entrar ya en el **if** de la **línea 80**

```
58 while (1) {  
59     copia_fdset(&rfd_a, &rfd, maximo+1);  
60     t_a = t;  
61     if ( (r = select(maximo+1, &rfd_a, NULL, NULL, &t_a)) < 0 ) {  
62         perror("select"); close(f2); close(f1); exit(-1);  
63     }  
64  
65     if (r == 0) {  
66         /* Se realiza la acción especificada. Ojo: read bloquearía */  
67     }  
68  
69     if (FD_ISSET(f1, &rfd_a)) {  
70         leidos = read(f1, b, T);  
71         if (leidos < 0) {  
72             perror("read"); exit(-1);  
73         } else if (leidos == 0) {  
74             FD_CLR(f1, &rfd);  
75         } else {  
76             /* proceso de la información recibida */  
77         }  
78     }  
79  
80     if (FD_ISSET(f2, &rfd_a)) {  
81         leidos = read(f2, b, T);  
82         if (leidos < 0) {  
83             perror("read"); exit(-1);  
84         } else if (leidos == 0) {  
85             FD_CLR(f1, &rfd);  
86         } else {  
87             /* proceso de la información recibida */  
88         }  
89     }  
90 }
```

select() y señales

- La función `select()`, mientras espera, también puede verse interrumpida por una señal
- Como ocurría con `read/write`, la función devuelve `-1` y asigna `errno = EINTR`
- Con esto, un programa que use `select()` y señales, debería gestionarlo:

```
7  while (1) {  
8      // usa una copia del fd_set original, porque select modifica el argumento  
9      copia_fdset(&conjunto_modificado,&conjunto_original, max_fd+1);  
10     //espera eventos indefinidamente  
11     resultado = select(max_fd + 1, &conjunto_modificado, NULL, NULL, NULL);  
12     if (resultado < 0) {  
13         if (errno == EINTR) {  
14             errno = 0;  
15             continue; //vuelve a iterar  
16         } else {  
17             perror("error en select");  
18             exit(1);  
19         }  
20     }
```

- La sentencia `continue` hace que el bucle vuelva a iterar y reintentar la llamada a `select()`

Más ejemplos con select()

- Espera la introducción de datos por teclado a la vez que se esperan datos de un descriptor de comunicaciones

```
FD_ZERO(&cjto_descriptores) ;
/* Incluimos en el select una fifo abierta*/
FD_SET(fd_fifo, &cjto_descriptores) ;
/* Incluimos en el select el descriptor de teclado */
FD_SET(0, &cjto_descriptores);
/*...*/
/*el primer argumento a select es el num. maximo de descriptor + 1 */
int resultado= select(fd_fifo+1, &cjto_modificado, NULL, NULL, NULL) ;
```

- Como forma de dormir un proceso con precisión de microsegundos, de forma portable entre familias Unix

```
struct timeval tv;
tv.tv_sec= 0;
tv.tv_usec=600;
select(0, NULL, NULL, NULL, &tv); /*duerme 600 microsegundos*/
```