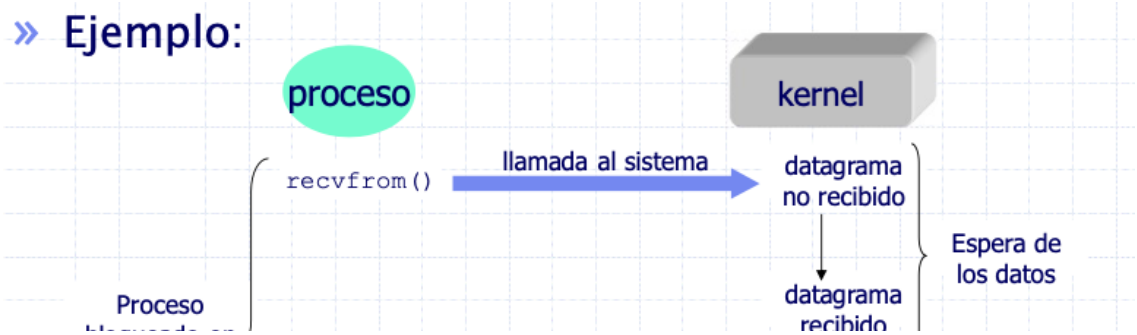
# Técnicas avanzadas de Programación en Redes

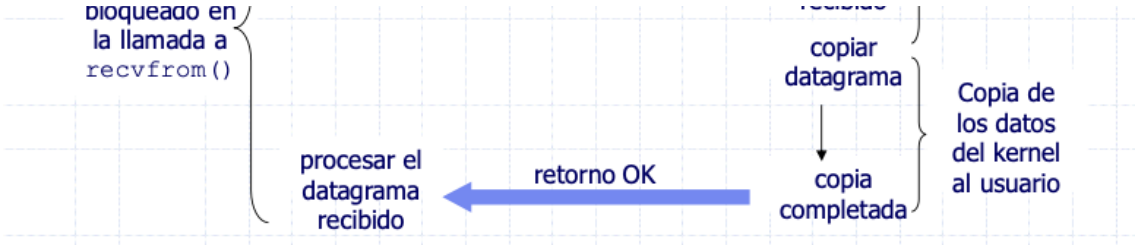
Entrada/Salida avanzada

- En una operación de entrada, como por ejemplo una lectura (read()), hay dos fases distintas a tener en cuenta:
  - Esperar a que los datos estén listos y copiar los datos desde el kernel a la memoria del proceso
- Igualmente, en una operación de salida (write()) hay también dos fases distintas a tener en cuenta:
  - esperar a que el destino de los datos, esté disponible para aceptarlos y
  - escribir los datos de la memoria del proceso en la memoria del kernel
- Cuál es el más importante
  - pico segundos para copiar
    - 0.25 * 10^9
    - 1 ciclo
  - milisegundos para ir a memoria
  - ssd micro segundos
    - erom
      - erasable read only memory
  - las esperas
    - para optimizar los r&w optimizar tiempos de esperas
    - llevar los datos de memoria a ram y leerlos lleva mucho tiempo
- cache es mejor ram
  - la cache es una ram chiquita y rápida
  - ram estática
    - cache
    - no refresca
  - ram dinámica
    - ram normal
- Una conexión de red cuanto puede ser el tiempo de espera
  - infinito / indefinido
- La red es mal lenta que la ram

E/S bloqueante

- La forma más simple de las operaciones de E/S que hemos visto es la forma bloqueante:
  - el proceso espera (queda bloqueado) a que la lectura o escritura se pueda llevar a cabo
- Atender a clientes simultaneamente al mismo tiempo





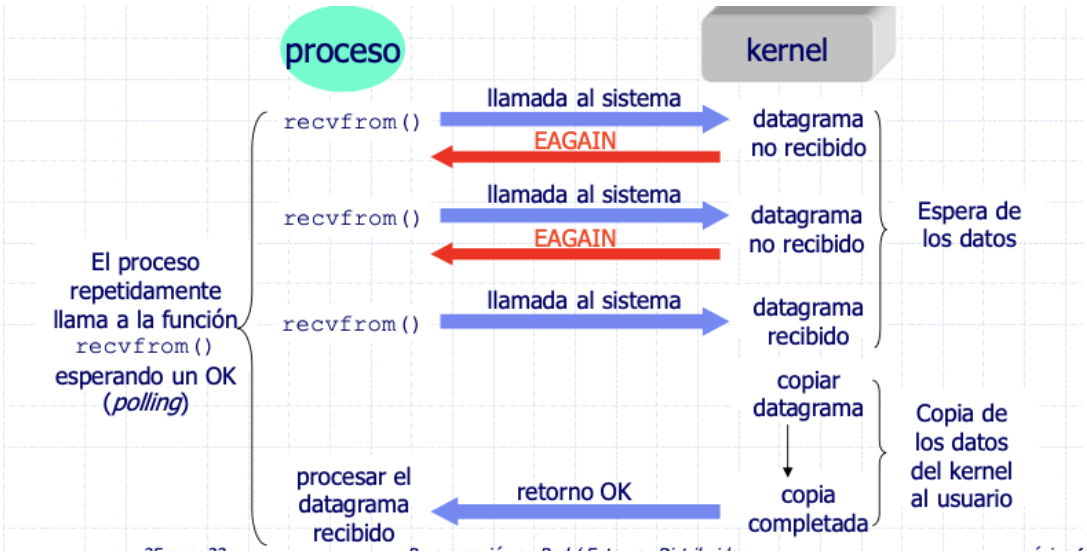

- Atenciones simultaneas no sirve
- tiempo de espera es lo que hay que atacar es lo que mas lleva

E/S avanzada

- El modelo de E/S bloqueante tiene varios inconvenientes
- Los más importantes son:
  - como puedo hacer lectura y escritura a la vez en el mismo descriptor (para soportar primitivas full - duplex)?
  - No podría hacer otra cosa mientras espero a que termine la operación de E/S?
  - Como leo de más de una fuente de datos a la vez?

E/S no bloqueante

- El modelo de E/S no bloqueante sirve para:
  - cuando una operación de E/S tenga que bloquear al proceso, no realice el bloqueo
  - Esta circunstancia se nos notifica como un error




- eagain
  - otra vez
- tarda en enviar el datagrama
- tarda en enviar la señal de error

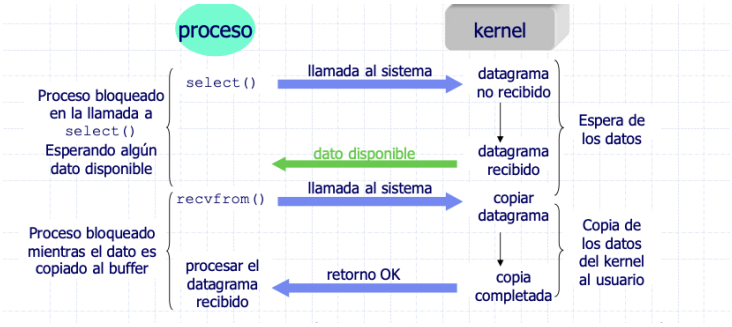
Interrogación periódica (polling)

- En el ejemplo anterior se está haciendo polling o interrogación periódica o muestreo
- El período de interrogación debería ser
  - pequeño para evitar demoras al usuario, pero
  - grande para evitar saturar a la cpu

- El polling es un desperdicio de un recurso de CPU y debería ser evitado en lo posible

E/S multiplexada

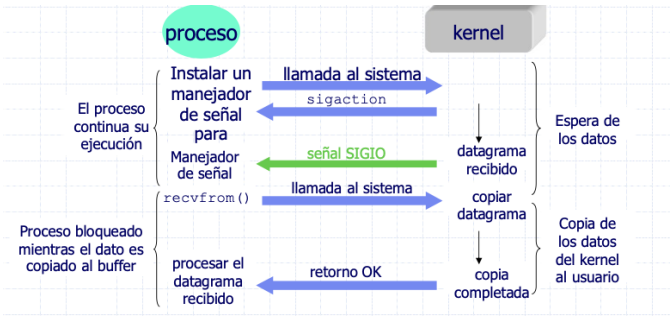
- El modelo de E/S multiplexada da la posibilidad de:
  - Construir una lista de descriptores en los que estamos interesados y
  - Llamar a una función que no vuelve hasta que uno de los descriptores esté listo para E/S




- En espera
  - nada

E/S conducida por señales

- En el modelo de E/S conducida por señales el kernel nos notifica con alguna señal (SIGIO o SIGPOLL) cuando una operación de E/S por la que un proceso esperaba está lista para ser realizada:




Tarda el mismo tiempo que select

- Tiempo de copia

Reentrante

- interrumpirse a si mismo

Señales

- mecanismo de cosas excepcionales
  - si mandamos muchas puede traer problemas
- usar señales y tratarlas es difícil

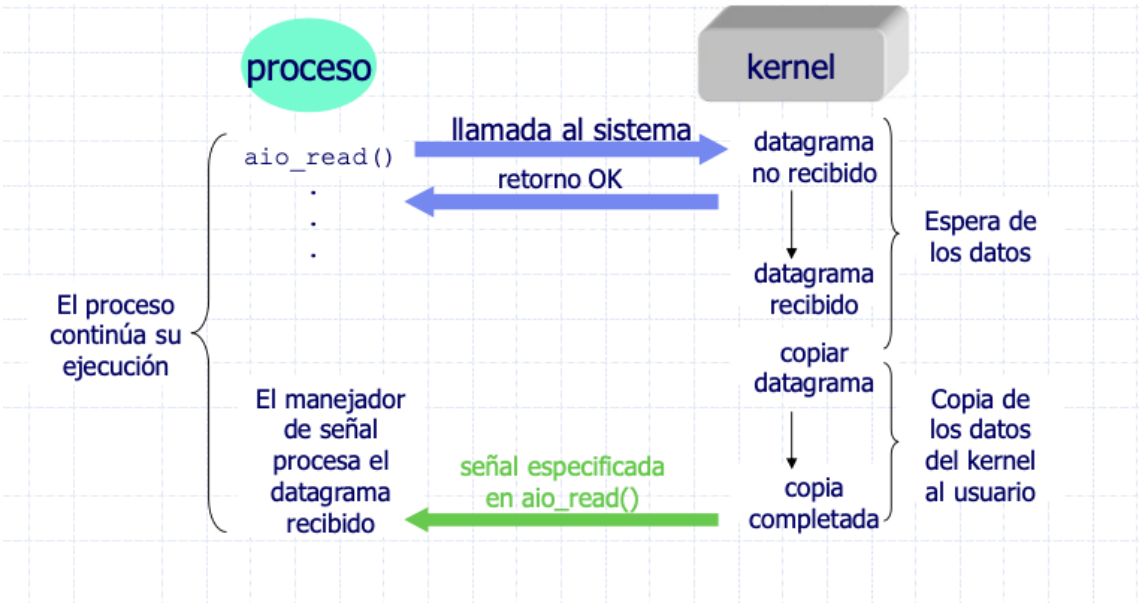
E/S Conducida por señales

- Independientemente de cómo manipulemos la señal, este sistema ofrece varias ventajas:
  - La ventaja fundamental de la E/S conducida por señales es que el proceso no permanece

bloqueado mientras espera a que los datos esten listos
- El programa puede continuar ejecutándose después de la petición
  - Y esperar a ser notificado por el manipulador de señales
  - o bien, que sea el propio manejador de señales el que realice la operación

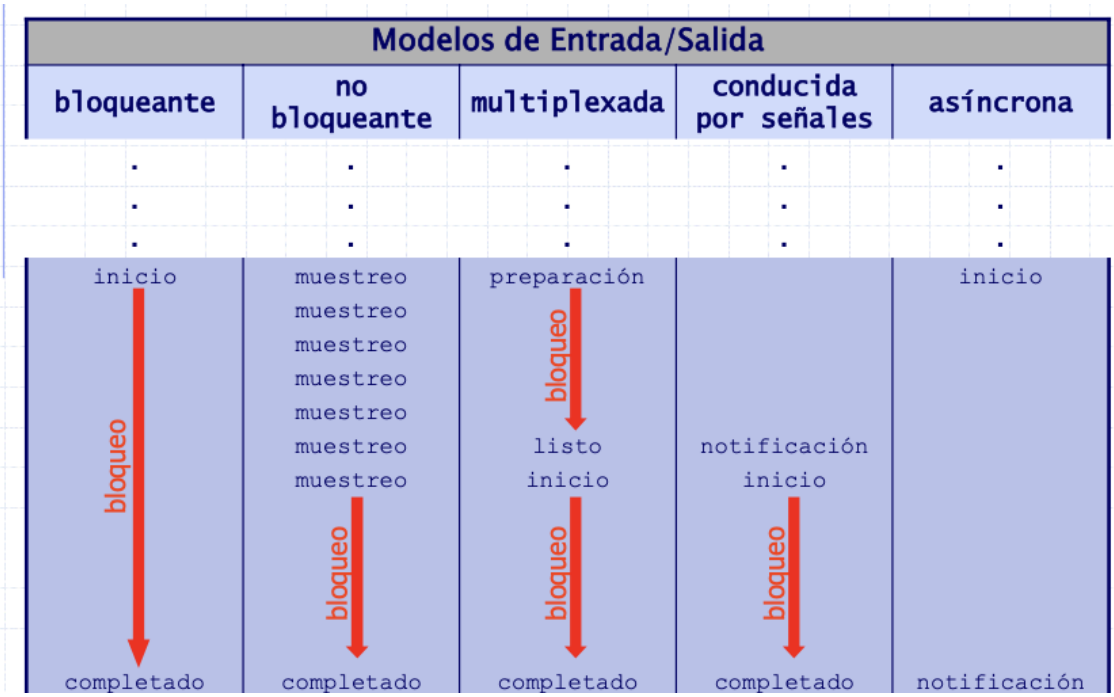
E/S Asíncrona
- El estándar POSIX 1. introdujo el modelo de E/S asíncrona
  - Las funciones del modelo asíncrono definidas por el estándar comienzan por los sufijos aio_ o lio_
  - En este esquema le indicamos al kernal el comienzo de una operación y éste nos notificará cuando se complete la operación entera, incluyendo el copiado de datos desde el kernel a la memoria del proceso
- La principal diferencia entre este modelo y la E/S conducida por señales es la siguiente
  - En este último el kernel nos avisa cuando una operación de e/s puede ser indicada.
  - en el modelo asíncrono el kernel nos indica cuando la operación está completada

El modelo de E/S asíncrona funciona de la siguiente forma




- callback
  - ya te avisare cuando este hecho

La primitiva aio_read() le pasa al kernel:
- el descriptor de E/S
- el puntero al buffer intermedio
- el tamaño del buffer
- un desplazamiento de fichero
- como nos notificará cuando la operación entera esté completa, seguramente, mediante algún tipo de señal
- Esta llamada al sistema retorna inmediatamente y nuestro proceso no es bloqueado por la espera de la operación de E/S




bloqueante

- el más fácil

Asíncrono

- es el mejor
- pero más difícil

Multiplexado

- segundo más fácil
- permite usar muchos clientes al mismo tiempo
- atender a un cliente simultaneamente

Normal y computación real?

- salir bien pero el tiempo no importa pero debería dar la misma respuesta
- tiempo real los datos tienen que salir rápido la respuesta puede ser diferente
  - no dar resultado adecuado
  - resultado cerca a la respuesta pero no la correcta

Multiplexada para atender a mucha gente
Asíncrona es primera división

- 100 millones al mismo servidor y todo quieren ver una peli
  - la web esta hecha con asíncrona
    - cursar mucho tráfico

Cambios de contexto

- cambiar de un proceso a otro
  - escrito en ensamblador

64 k  udp

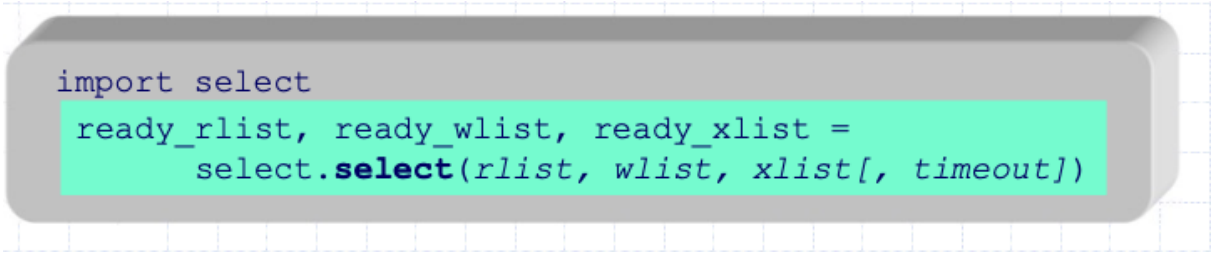
diff dos ficheros

E/S Multiplexada

- La E/S multiplexada se realiza mediante la primitiva select()
  - Se le pasa a select() una lista de descriptores de E/S
  - La llamada no devolverá control hasta que alguno de los descriptores de los dispositivos de entrada / salida no esté listo para su uso
  - La llamada nos devolverá los descriptores de los dispositivos que están listos para ser tratados
  - Ello puede ocurrir por tres causas
    - leer
    - escribir
    - obtener información de estado del dispositivo gestionado por el descriptor

La llamada select()

- A continuación se presenta la firma de select()
  - elementos válidos en las listas:
    - objetos de tipo fichero
      - los devuletos por open ()
    - Descriptores de ficheros
      - Los devueltos por os.open () o similares)
    - sockets (los devueltos por socket.socket)




- llamada al sistema
- system call

Cuando select esta bloqueado no podemos hacer nada
read es bloqueante pero no se bloquea la llamada porque sabe donde leer y ya estan ahi
si le pongo 0 permite muestrar

- Funcionamiento
  - En unix funciona con todo tipo de descriptores de fichero
  - En windows esta llamada solamente funciona con sockets ya que la implementa la librería winsock y no es el sistema de E/S
  - Es la llamada de muestreo por multiplataforma
  - timeout

        - voy a esperar 20 segundos
        - si ninguno me escribe después de 20 segundos te dice aqui tienes tu lista

- Select contesta que todo el mundo esta listo en la segunda
- No se usa la tercera lista
- Las otras dos van vacias solo importa rlist y el timeout

Parámetros de select( )

- Conjuntos de descriptores
    - listas de descriptores de E/S que contienen los dispositivos
        - rlist
            - desde los que el proceso desea leer datos
        - wlist
            - en los que el proceso desea escribir datos
        - xlist
            - desde los que el proceso desea estar informado de cambios excepcionales en los mismos
        - Si no se va a usar alguna lista (lo normal es usar solo rlist) puede pasarse como parámetro la lista vacía [ ]
    - Tiempo timeout:
        - límite superior del tiempo de espera de respuesta
            - hay tres posibilidades
            - Si tiempo de espera
                - muestreo retorno inmediato
                    - timeout = 0
            - Con tiempo de espera en segundos definido por el usuario
                - timeout = 4.5
                    - punto flotante, fracciones de segundo
                    - cota máxima
                    - devuelve a penas tenga para leer
            - Bloqueo (interrumpible mediante la recepción de una señal)
                - timeout = None
                    - valor por defecto

Valor de retorno de select()

- Valor de retorno de select( )
    - Devuelve tres listas con los descriptores listos para cada uno de los criterios
    - Cada una de las listas es un subjconjunto de las pasadas como parámetros
    - En windows no se garantiza que se soporte la operación con más de una lista
    - Si se produce un error (ejemplo la recepción de una señal)
        - select( ) lanza una excepción y fina errno con el valor apropiado

- En este caso los valores de retorno son indefinidos y no se debe confiar en ellos
  - posibles errores
    - EBADF
      - descriptor de fichero inválido en uno de los conjuntos
    - EINTR
      - capturada una señal no bloqueante
    - EINVAL
      - nfds es menor que cero
    - ENOMEM
      - no se ha podido pedir memoria para buffer interno

Filtrar quitar de la lista, si miro la lista original que yo pase es que desaparecieron cosas

- Le pasamos copias de la lista a select ya que la va a filtrar

Descriptores listos para su uso

- Cuándo está listo un descriptor
  - Un descriptor en el conjunto rlist está listo cuando es seguro que una lectura realizada sobre dicho descriptor no se bloqueará
  - Un descriptor es el conjunto wlist está listo cuando es seguro que una escritura realizada sobre dicho descriptor no se bloqueará
  - Un descriptor en el conjunto xlist está listo cuando
    - se reciben datos fuera de banda en una conexión de red
    - En determindas condiciones que peden darse en una pseudo terminal
  - Su uso queda como ejercicio para los más aventurados

Poll = encuesta / encuenstar

- muestrear

E/S Multiplexada

- SVR3 proporciona la llamada poll() solo para trabajar con identificadores de streams
  - Mecanismos que proporcionan conexiones full-duplex entre un proceso y un manejador de dispositivo de entrada / salida (i/o device driver )
- SVR4 la amplía para cualquier tipo de dispositivo

```
import select
objeto_poll = select.poll()
```

- Implementada en casi todas las versiones de Unix y algunos otros sistemas operativos (Windows)
  - Más eficiente n descriptores registrados 3 7 14 que la llamada select n de descriptor más alto 0-14

Interfaz de objetos poll( )

- obj_poll.register(fd[, eventmask])
  - Registrar el descriptor de fichero u objeto fichero fd
  - El parámetro opcional eventmask es la máscara de bit con el tipo de eventos sobre los que se muestrea a fd
    - Constantes a usar en eventmask
      - POLLIN datos pendientes de lectura
      - POLLPRI datos urgentes pendientes de lectura
      - POLLOUT la escritura no se bloqueará
      - POLLERR ha habido algún error
      - POLLHUP comunicación terminada (hung up)
      - POLLNVAL petición inválida (descriptor no abierto)
    - Valor por defecto POLLIN POLLPRI POLLOUT
  - obj_poll.modify(fd, eventmask)
    - Modifica un descriptor ya registrado con la nueva máscara
  - obj_poll.unregister(fd)
    - elimina el descriptor fd del registro
- [(fd, evento)] = obj_poll.poll([timeout)
  - Muestrea el conjunto de descriptores registrados
    - el parámetro timeout (es opcional) especifica un tiempo de espera
      - Si se pasa un valor, debe ser un entero en milisegundos
      - Si no se pasa, es cero, negativo, o none, la llamada se bloquea hasta que se produzca algún evento
  - Devuelve una lista de tuplas [(fd, evento)]
    - fd es el descriptor afectado
    - evento es la máscara de eventos para los que fd está listo
  - Una lista vacía indica que ha vencido el timeout sin que ningún descriptor este listo

Otras llamadas del módulo select

```
import select
objeto_epoll  = select.epoll([sizehint=-1])
objeto_kqueue = select.kqueue()
objeto_kevent = select.kevent(ident,filter=KQ_FILTER_READ,
                flags=KQ_EV_ADD, fflags=0, data=0, udata=0)
```

epoll()

- Sólo soportada en Linux

kqueue( ) y kevent ( )

- solo soportadas en bsd

Cada una de ellas devuelve un objeto diferente, similar al objeto poll, con una interfaz específica:

- Su uso queda como ejercicio para los más aventurados