**SOPER P1**

**Ejercicio 1**

1. **man -k pthread**

pthread\_attr\_destroy (3) - initialize and destroy thread attributes object

pthread\_attr\_getaffinity\_np (3) - set/get CPU affinity attribute in thread at...

pthread\_attr\_getdetachstate (3) - set/get detach state attribute in thread at...

pthread\_attr\_getguardsize (3) - set/get guard size attribute in thread attrib...

pthread\_attr\_getinheritsched (3) - set/get inherit-scheduler attribute in thr...

pthread\_attr\_getschedparam (3) - set/get scheduling parameter attributes in t...

pthread\_attr\_getschedpolicy (3) - set/get scheduling policy attribute in thre...

pthread\_attr\_getscope (3) - set/get contention scope attribute in thread attr...

pthread\_attr\_getstack (3) - set/get stack attributes in thread attributes object

pthread\_attr\_getstackaddr (3) - set/get stack address attribute in thread att...

pthread\_attr\_getstacksize (3) - set/get stack size attribute in thread attrib...

pthread\_attr\_init (3) - initialize and destroy thread attributes object

pthread\_attr\_setaffinity\_np (3) - set/get CPU affinity attribute in thread at...

pthread\_attr\_setdetachstate (3) - set/get detach state attribute in thread at...

pthread\_attr\_setguardsize (3) - set/get guard size attribute in thread attrib...

pthread\_attr\_setinheritsched (3) - set/get inherit-scheduler attribute in thr...

pthread\_attr\_setschedparam (3) - set/get scheduling parameter attributes in t...

pthread\_attr\_setschedpolicy (3) - set/get scheduling policy attribute in thre...

pthread\_attr\_setscope (3) - set/get contention scope attribute in thread attr...

pthread\_attr\_setstack (3) - set/get stack attributes in thread attributes object

pthread\_attr\_setstackaddr (3) - set/get stack address attribute in thread att...

pthread\_attr\_setstacksize (3) - set/get stack size attribute in thread attrib...

pthread\_cancel (3) - send a cancellation request to a thread

pthread\_cleanup\_pop (3) - push and pop thread cancellation clean-up handlers

pthread\_cleanup\_pop\_restore\_np (3) - push and pop thread cancellation clean-u...

pthread\_cleanup\_push (3) - push and pop thread cancellation clean-up handlers

pthread\_cleanup\_push\_defer\_np (3) - push and pop thread cancellation clean-up...

pthread\_create (3) - create a new thread

pthread\_detach (3) - detach a thread

pthread\_equal (3) - compare thread IDs

pthread\_exit (3) - terminate calling thread

pthread\_getaffinity\_np (3) - set/get CPU affinity of a thread

pthread\_getattr\_default\_np (3) - get or set default thread-creation attributes

pthread\_getattr\_np (3) - get attributes of created thread

pthread\_getconcurrency (3) - set/get the concurrency level

pthread\_getcpuclockid (3) - retrieve ID of a thread's CPU time clock

pthread\_getname\_np (3) - set/get the name of a thread

pthread\_getschedparam (3) - set/get scheduling policy and parameters of a thread

pthread\_join (3) - join with a terminated thread

pthread\_kill (3) - send a signal to a thread

pthread\_kill\_other\_threads\_np (3) - terminate all other threads in process

pthread\_mutex\_consistent (3) - make a robust mutex consistent

pthread\_mutex\_consistent\_np (3) - make a robust mutex consistent

pthread\_mutexattr\_getpshared (3) - get/set process-shared mutex attribute

pthread\_mutexattr\_getrobust (3) - get and set the robustness attribute of a m...

pthread\_mutexattr\_getrobust\_np (3) - get and set the robustness attribute of ...

pthread\_mutexattr\_setpshared (3) - get/set process-shared mutex attribute

pthread\_mutexattr\_setrobust (3) - get and set the robustness attribute of a m...

pthread\_mutexattr\_setrobust\_np (3) - get and set the robustness attribute of ...

pthread\_rwlockattr\_getkind\_np (3) - set/get the read-write lock kind of the t...

pthread\_rwlockattr\_setkind\_np (3) - set/get the read-write lock kind of the t...

pthread\_self (3) - obtain ID of the calling thread

pthread\_setaffinity\_np (3) - set/get CPU affinity of a thread

pthread\_setattr\_default\_np (3) - get or set default thread-creation attributes

pthread\_setcancelstate (3) - set cancelability state and type

pthread\_setcanceltype (3) - set cancelability state and type

pthread\_setconcurrency (3) - set/get the concurrency level

pthread\_setname\_np (3) - set/get the name of a thread

pthread\_setschedparam (3) - set/get scheduling policy and parameters of a thread

pthread\_setschedprio (3) - set scheduling priority of a thread

pthread\_sigmask (3) - examine and change mask of blocked signals

pthread\_sigqueue (3) - queue a signal and data to a thread

pthread\_spin\_destroy (3) - initialize or destroy a spin lock

pthread\_spin\_init (3) - initialize or destroy a spin lock

pthread\_spin\_lock (3) - lock and unlock a spin lock

pthread\_spin\_trylock (3) - lock and unlock a spin lock

pthread\_spin\_unlock (3) - lock and unlock a spin lock

pthread\_testcancel (3) - request delivery of any pending cancellation request

pthread\_timedjoin\_np (3) - try to join with a terminated thread

pthread\_tryjoin\_np (3) - try to join with a terminated thread

pthread\_yield (3) - yield the processor

pthreads (7) - POSIX threads

**b) man man**

2 Llamadas del sistema (funciones servidas por el núcleo)

**man 2 write**

**Ejercicio 2**

1. grep molino "don quijote.txt" >> "aventuras.txt"
2. ls | wc -l

you have to list all the files of the directory, then it count the number of lines there is in the ls command

1. cat "lista de la compra Pepe.txt" "lista de la compra Elena.txt" | sort | uniq | wc -l > "num compra.txt" 2> /dev/null

Utilizamos primero el comando cat para concatenar los dos archivos requeridos. Después ordenamos la lista con sort ya que el comando uniq, solo elimina las filas repetidas si están seguidas. Finalmente contamos el número de palabras con wc -l.

1. ps -A -L | awk '{print $1}' | uniq | wc -l > 'hilos.txt' /\*WE DNT KNOW IF ITS OKAY\*/

**Ejercicio 3**

1. No such file or directory. Corresponde al valor de errno 2.
2. Permission denied

Errno: 13

1. Guardamos el valor de errno en una variable auxiliar, después imprimimos el valor de errno con un printf o similar y reasignamos el valor que contiene la variable auxiliar a errno de nuevo. De este modo nos aseguramos que el mensaje de error de perrno pertenece al fopen previamente hecho.

**Ejercicio 4**

1. Se observa que el programa que el programa está utilizando toda la CPU.
2. Se puede ver que no consume recursos del CPU.

**Ejercicio 5**

1. En el caso de que el proceso no espere a los hilos, eliminando las líneas de código pthread\_exit, el programa termina, sin esperar a que los hilos terminen.
2. Si se elimina pthread\_join y se cambia exit por pthread\_exit, el programa crea los hilos, pero no los espera. Termina la función main y luego los threads siguen ejecutándose.
3. Para que habiendo eliminado pthread\_join(), funcione, deberíamos poner un pthread\_detach(). Esta función lo que hace es desligar los hilos, por lo que no tienen que esperar a otros hilos a que terminen.

**Ejercicio 6**

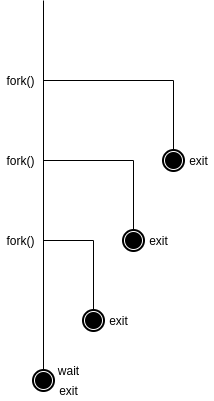
Archivo adjunto. (ejercicio\_hilos.c)

**Ejercicio 7**

1. No, porque depende del orden en el que se ejecuten los procesos.
2. else if (pid == 0){

printf (“hijo con PID %i\n”, getpid());

exit (EXIT\_SUCCESS);

}

1. Hay procesos que se quedan huérfanos ya que los hijos hacen un exit y esperan a que su padre los recoja. Como esto no pasa, ya que no se hace un wait en el proceso padre, se quedan todos huérfanos menos el último proceso hijo, que si que es recogido por el padre ya que fuera del bucle tiene un wait.
2. else if (pid == 0){

printf (“hijo con PID %i\n”, getpid());

exit (EXIT\_SUCCESS);

}else if (pid > 0){

printf(“Padre %i\n”, i);

wait (NULL);

}

Con la modificación que se acaba de hacer, cada vez que se hace fork, el padre espera a que su hijo termine de ejecutarse, por lo que no quedarían procesos hijos huérfanos.

**Ejercicio 8**

Archivo adjunto. (ejercicio\_arbol.c)

**Ejercicio 9**

1. Cuando se ejecuta el código no funciona. Esto ocurre por que cada vez que haces fork, cada uno crea su propio stack, que no es compartido. es por esto que si inicializas la string en el proceso hijo, no se guarda en el stack del proceso padre.
2. Hay que liberar memoria en el proceso padre y en el hijo, ya que al tener cada uno su propio stack, tiene que liberarse memoria individualmente.

**Ejercicio 10**

1. Archivo adjunto. (ejercico\_shell.c)
2. Se ha utilizado un execvp porque busca el fichero que se quiere ejecutar en los directorios de /bin/.
3. exercise 10:$ sh -c inexistente

sh: 1: inexistente: not found

1. Abortado (`core' generado)
2. Archivo adjunto. (ejercico\_shell\_spawn.c)

**Ejercicio 11**

1. Miramos el proceso 4068 utilizando el comando more status. Nombre: chrome.
2. /proc/4068
3. Utilizamos el comando more cmdline dentro del directorio del proceso.

/opt/google/chrome/chrome

1. echo $LANG

es\_ES.UTF-8

1. Para esto hay que acceder al directorio /task y ejecutar el comando ls. Al hacerlo en el proceso con pid 4068 obtenemos lo siguiente:

4068 4069 4071 4072 4073 4074 4075 4076 4077 4078 4707

**Ejercicio 12**

1. Tiene el descriptor abierto:

**ls -ld /proc/21916/fd**

dr-x------ 2 xiao xiao 0 feb 21 12:04

Apunta a los ficheros:

**ls -ld /proc/21916/fd/\***

lrwx------ 1 xiao xiao 64 feb 21 12:05 /proc/21916/fd/0 -> /dev/pts/0

lrwx------ 1 xiao xiao 64 feb 21 12:05 /proc/21916/fd/1 -> /dev/pts/0

lr-x------ 1 xiao xiao 64 feb 21 12:05 /proc/21916/fd/103 -> /usr/share/code/v8\_context\_snapshot.bin

lr-x------ 1 xiao xiao 64 feb 21 12:05 /proc/21916/fd/104 -> /usr/share/code/natives\_blob.bin

lrwx------ 1 xiao xiao 64 feb 21 12:05 /proc/21916/fd/12 -> 'anon\_inode:[eventpoll]'

lr-x------ 1 xiao xiao 64 feb 21 12:05 /proc/21916/fd/13 -> /usr/share/code/icudtl.dat

lr-x------ 1 xiao xiao 64 feb 21 12:05 /proc/21916/fd/14 -> /usr/share/code/v8\_context\_snapshot.bin

lr-x------ 1 xiao xiao 64 feb 21 12:05 /proc/21916/fd/15 -> /usr/share/code/natives\_blob.bin

lr-x------ 1 xiao xiao 64 feb 21 12:05 /proc/21916/fd/16 -> /usr/share/code/chrome\_100\_percent.pak

lr-x------ 1 xiao xiao 64 feb 21 12:05 /proc/21916/fd/17 -> /usr/share/code/chrome\_200\_percent.pak

lr-x------ 1 xiao xiao 64 feb 21 12:05 /proc/21916/fd/18 -> /usr/share/code/locales/es.pak

lr-x------ 1 xiao xiao 64 feb 21 12:05 /proc/21916/fd/19 -> /usr/share/code/resources.pak

lrwx------ 1 xiao xiao 64 feb 21 12:05 /proc/21916/fd/2 -> /dev/pts/0

lrwx------ 1 xiao xiao 64 feb 21 12:05 /proc/21916/fd/25 -> 'socket:[78684]'

lr-x------ 1 xiao xiao 64 feb 21 12:05 /proc/21916/fd/31 -> /usr/share/code/resources/app/node\_modules.asar

lrwx------ 1 xiao xiao 64 feb 21 12:05 /proc/21916/fd/32 -> 'socket:[79739]'

lrwx------ 1 xiao xiao 64 feb 21 12:05 /proc/21916/fd/33 -> '/dev/shm/.org.chromium.Chromium.23WPUF (deleted)'

lr-x------ 1 xiao xiao 64 feb 21 12:05 /proc/21916/fd/34 -> /usr/share/fonts/truetype/dejavu/DejaVuSansMono.ttf

lr-x------ 1 xiao xiao 64 feb 21 12:05 /proc/21916/fd/36 -> /usr/share/fonts/truetype/droid/DroidSansFallbackFull.ttf

lrwx------ 1 xiao xiao 64 feb 21 12:05 /proc/21916/fd/4 -> 'socket:[78109]'

lr-x------ 1 xiao xiao 64 feb 21 12:05 /proc/21916/fd/40 -> /usr/share/code/resources/electron.asar

lrwx------ 1 xiao xiao 64 feb 21 12:05 /proc/21916/fd/41 -> '/dev/shm/.org.chromium.Chromium.Lcrh19 (deleted)'

lrwx------ 1 xiao xiao 64 feb 21 12:05 /proc/21916/fd/42 -> '/dev/shm/.org.chromium.Chromium.U0vn6O (deleted)'

lrwx------ 1 xiao xiao 64 feb 21 12:05 /proc/21916/fd/43 -> '/dev/shm/.org.chromium.Chromium.7tuyEE (deleted)'

lr-x------ 1 xiao xiao 64 feb 21 12:05 /proc/21916/fd/44 -> /usr/share/fonts/truetype/ubuntu/Ubuntu-RI.ttf

lr-x------ 1 xiao xiao 64 feb 21 12:05 /proc/21916/fd/46 -> /usr/share/fonts/truetype/liberation2/LiberationSerif-Regular.ttf

lr-x------ 1 xiao xiao 64 feb 21 12:05 /proc/21916/fd/47 -> /usr/share/fonts/truetype/noto/NotoColorEmoji.ttf

lrwx------ 1 xiao xiao 64 feb 21 12:05 /proc/21916/fd/48 -> '/dev/shm/.org.chromium.Chromium.Ye3fkO (deleted)'

lrwx------ 1 xiao xiao 64 feb 21 12:05 /proc/21916/fd/49 -> '/dev/shm/.org.chromium.Chromium.qvEer1 (deleted)'

lrwx------ 1 xiao xiao 64 feb 21 12:05 /proc/21916/fd/5 -> 'socket:[80214]'

lrwx------ 1 xiao xiao 64 feb 21 12:05 /proc/21916/fd/50 -> '/dev/shm/.org.chromium.Chromium.5th1Hr (deleted)'

lrwx------ 1 xiao xiao 64 feb 21 12:05 /proc/21916/fd/51 -> '/dev/shm/.org.chromium.Chromium.Hr5nTE (deleted)'

lrwx------ 1 xiao xiao 64 feb 21 12:05 /proc/21916/fd/52 -> '/dev/shm/.org.chromium.Chromium.4lCL4R (deleted)'

lrwx------ 1 xiao xiao 64 feb 21 12:05 /proc/21916/fd/53 -> '/dev/shm/.org.chromium.Chromium.kN0ml4 (deleted)'

lrwx------ 1 xiao xiao 64 feb 21 12:05 /proc/21916/fd/54 -> '/dev/shm/.org.chromium.Chromium.jbuhg5 (deleted)'

lrwx------ 1 xiao xiao 64 feb 21 12:05 /proc/21916/fd/55 -> '/dev/shm/.org.chromium.Chromium.drnSri (deleted)'

lr-x------ 1 xiao xiao 64 feb 21 12:05 /proc/21916/fd/60 -> /usr/share/fonts/truetype/liberation2/LiberationSans-Italic.ttf

lrwx------ 1 xiao xiao 64 feb 21 12:05 /proc/21916/fd/62 -> '/dev/shm/.org.chromium.Chromium.FtXADv (deleted)'

lr-x------ 1 xiao xiao 64 feb 21 12:05 /proc/21916/fd/63 -> /usr/share/fonts/truetype/dejavu/DejaVuSans.ttf

lr-x------ 1 xiao xiao 64 feb 21 12:05 /proc/21916/fd/64 -> /usr/share/fonts/truetype/ubuntu/Ubuntu-R.ttf

lr-x------ 1 xiao xiao 64 feb 21 12:05 /proc/21916/fd/65 -> /usr/share/fonts/truetype/liberation2/LiberationSans-Regular.ttf

lr-x------ 1 xiao xiao 64 feb 21 12:05 /proc/21916/fd/66 -> /usr/share/fonts/truetype/ubuntu/Ubuntu-B.ttf

lr-x------ 1 xiao xiao 64 feb 21 12:05 /proc/21916/fd/67 -> /usr/share/fonts/truetype/liberation2/LiberationSans-Bold.ttf

lrwx------ 1 xiao xiao 64 feb 21 12:05 /proc/21916/fd/68 -> '/dev/shm/.org.chromium.Chromium.TxzkPI (deleted)'

lrwx------ 1 xiao xiao 64 feb 21 12:05 /proc/21916/fd/69 -> '/dev/shm/.org.chromium.Chromium.lcwb1V (deleted)'

lrwx------ 1 xiao xiao 64 feb 21 12:05 /proc/21916/fd/70 -> '/dev/shm/.org.chromium.Chromium.lBs7c9 (deleted)'

lrwx------ 1 xiao xiao 64 feb 21 12:05 /proc/21916/fd/72 -> '/dev/shm/.org.chromium.Chromium.z383Az (deleted)'

lr-x------ 1 xiao xiao 64 feb 21 12:05 /proc/21916/fd/75 -> /usr/share/fonts/truetype/ubuntu/Ubuntu-L.ttf

lrwx------ 1 xiao xiao 64 feb 21 12:05 /proc/21916/fd/79 -> '/dev/shm/.org.chromium.Chromium.ofo4om (deleted)'

lrwx------ 1 xiao xiao 64 feb 21 12:05 /proc/21916/fd/80 -> '/dev/shm/.org.chromium.Chromium.xuR6MM (deleted)'

lrwx------ 1 xiao xiao 64 feb 21 12:05 /proc/21916/fd/81 -> '/dev/shm/.org.chromium.Chromium.nZMdZZ (deleted)'

lrwx------ 1 xiao xiao 64 feb 21 12:05 /proc/21916/fd/82 -> '/dev/shm/.org.chromium.Chromium.QPIlbd (deleted)'

lrwx------ 1 xiao xiao 64 feb 21 12:05 /proc/21916/fd/83 -> '/dev/shm/.org.chromium.Chromium.ovbBnq (deleted)'

lrwx------ 1 xiao xiao 64 feb 21 12:05 /proc/21916/fd/84 -> '/dev/shm/.org.chromium.Chromium.ujGRzD (deleted)'

lrwx------ 1 xiao xiao 64 feb 21 12:05 /proc/21916/fd/85 -> '/dev/shm/.org.chromium.Chromium.8LzdMQ (deleted)'

1. Stop 2: Crea un archivo con descripción de fichero salida estándar

lrwx------ 1 xiao xiao 64 feb 21 12:08 /proc/21916/fd/3 -> '/home/xiao/Escritorio/Ing. Inf II/2 Cuatri/SOPER/P1/Práctica1/file1.txt'

Stop 3 Crea un archivo con descripción de fichero salida estándar

lrwx------ 1 xiao xiao 64 feb 21 12:09 /proc/21916/fd/6 -> '/home/xiao/Escritorio/Ing. Inf II/2 Cuatri/SOPER/P1/Práctica1/file2.txt'

1. Ahora te aparece como borrado el fichero 1 ya que en el código se utiliza un “unlink”, que hace que se borre la ruta para acceder al fichero.

lrwx------ 1 xiao xiao 64 feb 21 12:08 /proc/21916/fd/3 -> '/home/xiao/Escritorio/Ing. Inf II/2 Cuatri/SOPER/P1/Práctica1/file1.txt (deleted)'

De todas maneras, se puede seguir accediendo a él, ya que sólo se borra del disco cuando todas las rutas que se usan para acceder a él sean borradas.

Utilizando el comando cat si que se puede ver el contenido del fichero guardado como 3.

**cat 3**

Hello

1. Stop 5 Elimina completamente el file 1

Stop 6 Crea el file 3 guardado como 3

lrwx------ 1 xiao xiao 64 feb 21 12:24 /proc/21916/fd**/3** -> '/home/xiao/Escritorio/Ing. Inf II/2 Cuatri/SOPER/P1/Práctica1/file3.txt'

Stop 7 Hace una copia del file 3, guardado ahora como 7

lr-x------ 1 xiao xiao 64 feb 21 12:24 /proc/21916/fd**/7** -> '/home/xiao/Escritorio/Ing. Inf II/2 Cuatri/SOPER/P1/Práctica1/file3.txt'

Se puede deducir el fichero que se está utilizando utilizando a través de la tabla de descriptores utilizando este como índice para la tabla.

**Ejercicio 13**

1. Escribe dos veces “Yo soy tu padre”. Una antes del “No” y otra después.

Esto ocurre ya que imprime el “Yo soy tu padre” antes de hacer fork. Cuando lo hace, guarda la string como padre, imprime el hijo y luego, como padre, vuelve a imprimir la string.

1. Al incluir \n al final del primer mensaje, éste solo aparece una vez.

Se debe a que al escribir \n vacía el buffer directamente imprimiendolo en stdout.

1. Al utilizar fprintf hacia un fichero nuevo, el mensaje “Yo soy tu padre” continúa apareciendo dos veces independientemente de que incluyamos \n o no. El mensaje de “No” aparece después de que se hayan escrito los dos mensajes anteriores.

Esto pasa porque al utilizar printf estábamos utilizando la salida estándar, donde si ponemos el \n se vacía el buffer, cosa que no ocurre si utilizamos fprintf con una salida hacia un fichero.

1. Si queremos que solo lo imprima una vez, tan solo hay que añadir el printf dentro de un else if como padre.

else if (pid > 0){

printf(“Yo soy tu padre”);

wait(NULL);

exit(EXIT\_SUCCESS);

}

**Ejercicio 14**

1. He recibido el string: Hola a todos! /\*Se imprime la string que el padre lee del hijo.\*/

He escrito en el pipe /\*El hijo escribe en el padre una string y se la manda a través del comando write\*/

Puede también imprimir primero la segunda línea (línea del hijo) y luego la primera línea (línea del padre), ya que el cómo se imprima depende del orden en el que se ejecuten los procesos.

He escrito en el pipe

He recibido el string: Hola a todos!

1. Que no termina el programa. Esto se debe a que antes de cada lectura, el programa cierre el extremo de escritura, para indicar que se ha terminado de escribir.
2. Se imprime siempre de esta manera:

He escrito en el pipe

He recibido el string: Hola a todos!

Es debido a que al poner un sleep en el hijo, da igual en qué orden se ejecuten los procesos, el padre tendrá que esperar a que el hijo termine, por lo que éste ejecutará su comando antes. Si no se pone “sleep(1)”, el programa ejecuta el padre y el hijo al mismo tiempo, y no se puede determinar si va a imprimir primero el padre o el hijo.

**Ejercicio 15**

Archivo adjunto. (ejercico\_pipes.c)