Práctica 1: Shell, Procesos, ficheros y tuberías

Autores: Marcos Aarón Bernuy, Kevin de la Coba Malam. Pareja 04 del grupo 2292.

**Ejercicio 1: Uso del manual.**

1. *Buscar en el manual la lista de funciones disponibles para el manejo de hilos y copiarla en la memoria junto con el comando usado para mostrarla. Las funciones de manejo de hilos comienzan por “pthread”.*

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

pthread\_attr\_getaffinity\_np (**3**) - set/get CPU affinity attribute in **thread** attributes object

pthread\_attr\_getdetachstate (**3**) - set/get detach state attribute in **thread** attributes object

pthread\_attr\_getguardsize (**3**) - set/get guard size attribute in **thread** attributes object

pthread\_attr\_getinheritsched (**3**) - set/get inherit-scheduler attribute in **thread** attributes object

pthread\_attr\_getschedparam (**3**) - set/get scheduling parameter attributes in **thread** attributes object

pthread\_attr\_getschedpolicy (**3**) - set/get scheduling policy attribute in **thread** attributes object

pthread\_attr\_getscope (**3**) - set/get contention scope attribute in **thread** attributes object

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

pthread\_attr\_getstackaddr (**3**) - set/get stack address attribute in **thread** attributes object

pthread\_attr\_getstacksize (**3**) - set/get stack size attribute in **thread** attributes object

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

pthread\_attr\_setaffinity\_np (**3**) - set/get CPU affinity attribute in **thread** attributes object

pthread\_attr\_setdetachstate (**3**) - set/get detach state attribute in **thread** attributes object

pthread\_attr\_setguardsize (**3**) - set/get guard size attribute in **thread** attributes object

pthread\_attr\_setinheritsched (**3**) - set/get inherit-scheduler attribute in **thread** attributes object

pthread\_attr\_setschedparam (**3**) - set/get scheduling parameter attributes in **thread** attributes object

pthread\_attr\_setschedpolicy (**3**) - set/get scheduling policy attribute in **thread** attributes object

pthread\_attr\_setscope (**3**) - set/get contention scope attribute in **thread** attributes object

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

pthread\_attr\_setstackaddr (**3**) - set/get stack address attribute in **thread** attributes object

pthread\_attr\_setstacksize (**3**) - set/get stack size attribute in **thread** attributes object

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-up handlers **while** saving cancelability type

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 handlers **while** saving cancelability type

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 mutex attributes object

pthread\_mutexattr\_getrobust\_np (**3**) - get and set the robustness attribute of a mutex attributes object

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

pthread\_mutexattr\_setrobust (**3**) - get and set the robustness attribute of a mutex attributes object

pthread\_mutexattr\_setrobust\_np (**3**) - get and set the robustness attribute of a mutex attributes object

pthread\_rwlockattr\_getkind\_np (**3**) - set/get the read-write lock kind of the **thread** read-write lock attribute object

pthread\_rwlockattr\_setkind\_np (**3**) - set/get the read-write lock kind of the **thread** read-write lock attribute object

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\_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 threadspthread\_testcancel (**3**) - request delivery of any pending cancellation request

Comando: man -k pthread > pthread.txt

Con el commando man -k pthread buscamos toda la información en relación a pthread, al añadir el ‘>’ redirigimos la salida de esta información al archivo “pthread.txt”.

1. *Consultar en la ayuda en qué sección del manual se encuentran las “llamadas al sistema” y buscar información sobre la llamada al sistema write. Escribir en la memoria los comandos usados.*

Si ejecutamos man manpodemos ver que las llamadas al sistema están en la sección 2 del manual. Si usamos man 2 write, obtenemos toda la información con respecto las llamadas al sistema de write.

**Ejercicio 2: Comandos y redireccionamiento.**

1. *Escribir un comando que busque las líneas que contengan “molino” en el fichero “don quijote.txt” y las añada al final del fichero “aventuras.txt”. Copiar el comando en la memoria, justificando las opciones utilizadas.*

Grep molino don\ \quijote.txt >> aventuras.txt

Grep se encarga de buscar la palabra molino que exista dentro del archivo aventuras.txt, luego con “>>” redirigimos la salida al final del archivo de aventuras.txt, básicamente hacemos un “apend”.

1. *Elaborar un pipeline que cuente el número de ficheros en el directorio actual. Copiar el pipeline en la memoria, justificando los comandos y opciones utilizados.*

Ls | wc -l. ls nos devuelve los archivos, y wc -l cuenta las líneas.

1. *Elaborar un pipeline que cuente el número de líneas distintas al concatenar “lista de la compra Pepe.txt” y “lista de la compra Elena.txt” y lo escriba en “num compra.txt”. Si alguno de los ficheros no existe, hay que ignorar los mensajes de error, para lo cual se redirigiría la salida de errores a /dev/null. Copiar el pipeline en la memoria, justificando los comandos y opciones utilizados.*

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

Con el comando cat concatenamos los dos archivos, luego con 2> /dev/null en caso de error guardamos la salida ahí , hacemos un pipe para que con sort -u cojamos las líneas únicas, hacemos otro pipe con wc -l para contar el número de líneas, y por último usamos > num\ \compra.txt para redirigir la salida al archivo.

**Ejercicio 3: Control de Errores.** Escribir un programa que abra un fichero indicado por el primer parámetro en modo lectura usando la función fopen. En caso de error de apertura, el programa mostrara el mensaje de error correspondiente por pantalla usando perror.

1. *¿Qué mensaje se imprime al intentar abrir un fichero inexistente? ¿A qué valor de errno corresponde?*

"No such file or directory", el valor de errno = 2.

1. *¿Qué mensaje se imprime al intentar abrir el fichero /etc/shadow? ¿A qué valor de errno corresponde?*

"Permission denied", el valor de errno = 13

1. *Si se desea imprimir el valor de errno antes de la llamada a perror, ¿qué modificaciones se deberían realizar para garantizar que el mensaje de perror se corresponde con el error de fopen?*

Guardamos errno en una variable y tras la llamada reasignamos el valor de errno.

**Ejercicio 4: Espera activa e inactiva.**

1. *Escribir un programa que realice una espera de 10 segundos usando la función clock en un bucle. Ejecutar en otra terminal el comando top. ¿Qué se observa?*

Al hacer el comando top el proceso que ejecuta clock, ocupa un 100% de la cpu.

1. *Reescribir el programa usando sleep y volver a ejecutar top. ¿Ha cambiado algo?*

Sí, el proceso no aparece en top. No ocupa nada en el procesador.

**Ejercicio 5: Finalización de Hilos.**

1. *¿Qué hubiera pasado si el proceso no hubiera esperado a los hilos? Para probarlo basta eliminar las llamadas a pthread\_join.*

Los hilos secundarios no les da tiempo a escribir. La salida es aleatoria ya que depende del planificador, pero en general los hilos secundarios no son capaces de escribir su mensaje.

1. *Con el código modificado del apartado anterior, indicar qué ocurre si se reemplaza la función exit por una llamada a pthread\_exit.*

Los otros hilos siguen ejecutándose aun habiendo acabado el hilo principal.

1. *Tras eliminar las llamadas a pthread\_join en los apartados anteriores, el programa es ahora incorrecto porque no se espera a que terminen todos los hilos. Escribir en la memoria el código que sería necesario añadir para que sea correcto no esperar a los hilos creados.*

...

pthread\_detach(h1);

...

pthread\_detach(h2);

Desligamos a los hilos del hilo principal para que sean independientes. Esto se ejecuta jsto después de crear cada hilo.

SEMANA2-------------------------------------------------------------------------------

SEMANA3-------------------------------------------------------------------------------

**Ejercicio 13: Shell**. Escribir un programa en C (“proc\_shell.c”) que implemente una shell sencilla (sin redirecciones ni estructuras de control). El ejercicio se divide en cuatro partes diferenciadas.

c) *Análisis de ejecución:*

* *Explicar qué función de la familia exec se ha usado y por qué. ¿Podría haberse usado otra? ¿Por qué?*

Hemos usado la función execvp ya que no hace falta que pongamos la dirección entera del comando. Por ejemplo, al usar ls, si usásemos execv tendríamos que poner /bin/ls, esto responde a la siguiente pregunta, sí, sí se puede usar otra, también tenemos otras posibilidades que actúan como execve.

* *Ejecutar con la shell implementada el comando sh -c inexistente. ¿Qué contiene la cadena con el resultado de finalización?*

sh -c inexistente tiene como salida: sh: 1: inexistente: not found

* *Hacer un programa en C que finalice llamando a abort y ejecutarlo con la shell implementada. ¿Qué contiene la cadena en este otro caso?*

Terminated by signal 6