

# Sistemas Operativos. Práctica 1.

Pablo Cuesta Sierra y Álvaro Zamanillo Sáez

## Semana 1

### Ejercicio 1

a) Para buscar las funciones relacionadas con hilos, busquemos todas aquellas que contengan “pthread” usando la opción -k en el comando man. El comando a usar es:

```
$ man -k pthread
```

Lista de funciones resultante:

```
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_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) Consultar en el manual en qué sección se encuentran las llamadas a sistema y buscar información sobre write:

```
$ man man
```

Con este comando averiguamos que la sección relacionada a *system calls* es la 2. Por lo tanto usamos el comando:

```
$ man 2 write
```

## Ejercicio 2

a) El comando empleado es:

```
$ grep -w molino "don quijote.txt" >> aventuras.txt
```

Usamos el comando grep para buscar las apariciones de “molino”. Como queremos que sea la palabra *molino* y no el grama *molino*, añadimos la opción -w. Finalmente redireccionamos la salida con >> en vez de > para que se añada al final del fichero, en lugar de reemplazarlo.

b) El *pipeline* es el siguiente:

```
$ ls | wc -l
```

La salida de ls es una lista (fichero) con los ficheros del actual directorio. Esta lista la usamos como input del comando wc, que acompañado de la opción -l, cuenta las líneas de dicha lista.

c) En este caso el *pipeline* es:

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

Primero concatenamos los dos ficheros (redirigiendo el mensaje error, en caso de haberlo). Después, dirigimos la salida para que sea el input de `sort`, para que luego `uniq` quite las líneas repetidas. Finalmente, contamos el número de líneas distintas con `wc -l` y dirigimos la salida al fichero `"num compra.txt"`.

### Ejercicio 3

- a) Si intentamos abrir un fichero inexistente recibimos el mensaje “No such file or directory”. El código `errno` asociado es 2.
- b) En el caso de intentar abrir el fichero `/etc/shadow` el mensaje de error es “Permission denied”, que corresponde al valor de `errno` 13.
- c) Justo después de la instrucción `fopen()` se debería guardar el valor de `errno` en otra variable ya que la llamada a `perror()` podría modificar la variable global `errno`.

```
pf=fopen(args[1], "r");
x=errno;
printf("El valor de errno es %i",x);
errno=x; /*Aquí aseguramos que la función perror() va a imprimir
el código de error de fopen ya que hemos restaurado el valor
de errno asociado a fopen()*/
perror();
```

### Ejercicio 4

- a) Durante los 10 segundos de espera el proceso asociado al programa está en estado “R” (runnable), es decir se está ejecutando (intercalándose con otros).
- b) En este caso el proceso está en estado “S” (interruptible sleep) o lo que es lo mismo, esperando a un suceso (el fin de la espera marcada por `sleep()`), mientras se ejecutan otros procesos.

### Ejercicio 5