Presentación

Objetivos

Esta práctica tiene como objetivo el diseño e implementación de una aplicación concurrente multi-hilo utilizando phtreads. Para su realización se pondrán en práctica muchos de los conceptos presentados en esta asignatura.

Presentación de la práctica

Hay que presentar los archivos con el código fuente realizado. Toda la codificación se hará exclusivamente en lenguaje C.

Enunciado de la práctica

El objetivo de la práctica es implementar una versión concurrente de un servidor de transferencia de ficheros (ftp) concurrente.

Se os proporciona la versión secuencial del cliente y del servidor ftp y vosotros tenéis que modificar el servidor para que cada una de las conexiones/sesiones de los usuarios (y sus comandos respectivos) se puedan realizar de forma concurrente, utilizando threads.

El además de atender la recepción de las peticiones y crear un thread para ejecutarlas, debe mostrar una estadísticas parciales (cada vez que un cliente se desconecta) y totales (al finalizar el servidor). Entre estás estadísticas que se tiene que calcular tenemos el tiempo de duración de las sesiones de los usuarios, el número de comandos ejecutados totales y por minuto, número de ficheros subidos (put) y bajados (get) y la transferencias promeido (MB/s) de subida y bajada. Está información la tiene que devolver el thread cuando el usuario finaliza la sesión (exit).

También es necesario asegurarse que si el servidor es finalizado (mediante un control+C), antes de morir se tiene que cancelar todos los threads de las sesiones abiertas de los usuarios. Los hilos se tienen que finalizar de forma controlada.

Al servidor concurrente se le pasará los mismos parámetros que a la versión secuencial más un parámetro adicional para definir el número de máximo de peticiones/threads concurrentes que se permite:

Sintaxis: `siftpdc <directorio_ftp> <port number> <Max_Threads>`

Funciones involucradas.

En la práctica podéis utilizar las siguientes funciones de c, entre otras:

- ```
  int pthread_create(pthread_t *thread, const pthread_attr_t *attr,
                     void *(*start_routine) (void *), void *arg);
  ```
- ```
  int pthread_join(pthread_t thread, void **retval);
  ```

- `void pthread_exit(void *retval);`
- `int pthread_cancel(pthread_t thread);`

Análisis prestaciones

Calcular el tiempo de ejecución de la versión concurrente y compararlo con el de la versión secuencial. Discutir los resultados obtenidos. El objetivo de la práctica es que **la versión concurrente proporcione un mejor rendimiento que la versión secuencial**.

Analizar también, cómo evoluciona el tiempo de ejecución en función del número de hilos utilizados en la aplicación concurrente. Entregar un pequeño informe en donde se muestren (preferiblemente de forma gráfica) y expliquen los resultados obtenidos.

Para obtener estos resultados, posiblemente necesitéis ampliar el teste de pruebas, para enviar más trabajo al servidor.

Indicar en el informe las características del hardware en donde habéis obtenido los resultados (especialmente el número de procesadores/núcleos).

Evaluación

La evaluación de la práctica se realizará en función del grado de eficiencia/concurrencia logrado.

Se utilizarán los siguientes criterios a la hora de evaluar las dos implementaciones de la práctica, partiendo de una nota base de 5:

- Aspectos que se evalúan para cada una de las versiones:
  - Ejecución Secuencial (Susp)
  - Ejecución no determinista ([-1.0p,-5.0p])
  - Prestaciones Servidor (-1.0p, +0.5p)
  - No mejora el tiempo de la versión secuencial (-2.0p)
  - Número incorrecto de hilos (-0.5p)
  - Join hilos (-0.5p)
  - Control Errores: cancelación hilos (-0.5p)
  - Condiciones de carrera (-0.5p)
  - No liberar memoria: (-0.25p,-0.5p)
  - Descomposición desbalanceada (Versión óptima) (-0.5p)
  - Descomposición parcial: falta asignar resto división (-0.5p)
- Evaluación del informe de la práctica:

  + Informe completo, con gráficos y conclusiones correctas (+0.25p,+1.0p)

- Sin informe (-1.0p)
- Informe con incoherencias en los tiempos sin justificar (-0.5p)
- Informe: análisis poco riguroso (-0.5p)
- No se explica el diseño de la solución propuesta (-0.5p)
- Descripción de los problemas encontrados y como se han solventado.

• Estilo del código.

- Comentarios
- Utilizar una correcta identación
- Descomposición Funcional (Código modular y estructurado)
- Control de errores.
- Test unitarios para verificar la correcta ejecución de la lógica de la aplicación.

Se puede tener en cuenta criterios adicionales en función de la implementación entregada.

Versiones Secuenciales

Junto con el enunciado se os proporcionamos una versión secuencial en C de un cliente y servidor de FTP simple[1]. Podéis utilizar dichas versiones para implementar las versiones concurrentes que se os pide en esta práctica.

La versión secuencial consta de dos programas el cliente (siftp) y el servidor (siftpd). El servidor soporta los comandos típicos de ftp: cd, ls, put, get, exit. La contraseña para acceder al servidor es "*scp18*".

La compilación del servidor se realiza mediante el makefile adjuntado (make all).

Para ejecutar el cliente hay que especificar dos parámetros: la ip de la máquina en donde se ejecuta el servidor y el puerto en donde recibe las peticiones de los clientes:

siftp <IP servidor> <número puerto>

Por ejemplo:

- siftp localhost 77777

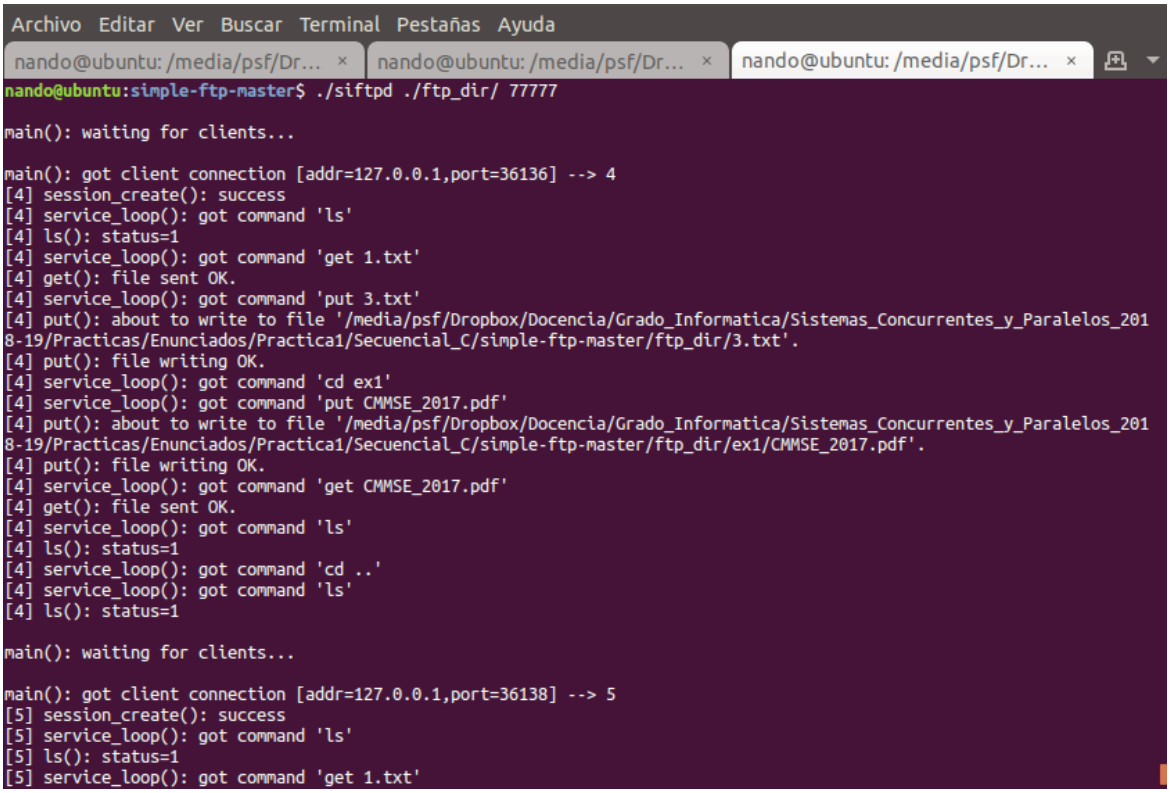
Para ejecutar el servidor hay que especificar dos parámetros: el path del directorio en donde se ubican los archivos que puede acceder los usuarios y el puerto por el cueal va a recibir las peticiones de los clientes:

siftpd <directorio_ftp> <número puerto>

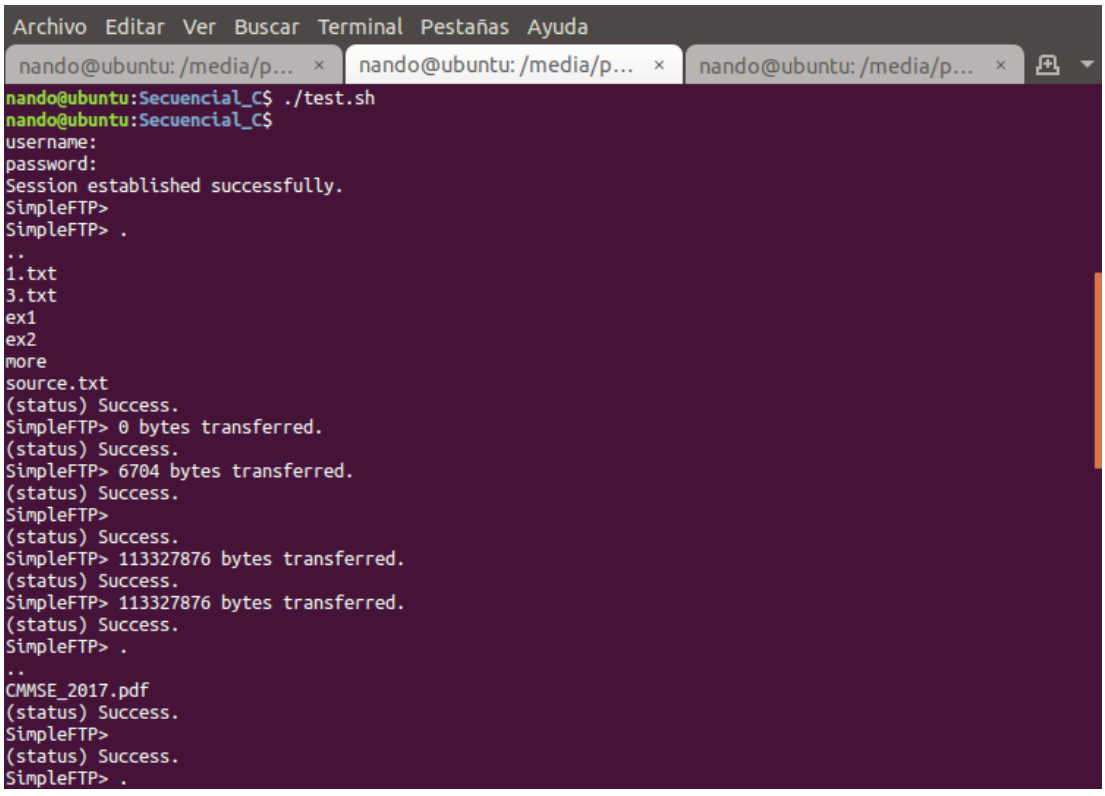
Por ejemplo:

[1] Opcionalmente, también se puede utilizar el servidor implementado en redes el año pasado para desarrollar la versión concurrente. En este caso, antes de daros el permiso para utilizar dicho servidor, tenéis que enviarme el código secuencial que implementasteis el curso pasado.

- siftpd ./ftp_dir 77777



Se os adjunta también varios ficheros de test para poder simular el comportamiento de los múltiples clientes concurrentes. Podéis modificar dichos scripts para incrementar la cargar desde el punto de vista del número de clientes concurrentes y el volumen de ficheros subidos y bajados.



Formato de entrega

MUY IMPORTANTE: La entrega de código que no compile correctamente, implicará suspender TODA la práctica.

No se aceptarán prácticas entregadas fuera de plazo (salvo por razones muy justificadas).

La entrega presencial de esta práctica es obligatoria para todos los miembros del grupo.

Comenzar vuestros programas con los comentarios:

```
/* ------------------------------------------------------------------
Práctica 1.
Código fuente : gestor.c
Grau Informàtica
NIF i Nombre completo autor1.
NIF i Nombre completo autor2.
------------------------------------------------------------------ */
```

Para presentar la práctica dirigiros al apartado de Actividades del Campus Virtual de la asignatura de Sistemas Concurrentes y Paralelos, ir a la actividad "Práctica 1" y seguid las instrucciones allí indicadas.

Se creará un fichero tar con todos los ficheros fuente de la práctica, con el siguiente comando:

```
$ tar -zcvf prac1.tgz fichero1 fichero2 ...
```

se creará el fichero "prac1.tgz" en donde se habrán empaquetado y comprimido lso ficheros "fichero1", fichero2, y ...

Para extraer la información de un fichero tar se puede utilizar el comando:

```
$ tar -zxvf tot.tgz
```

El nombre del fichero tar tendrá el siguiente formato: "Apellido1Apellido2PRA2.tgz". Los apellidos se escribirán sin acentos. Si hay dos autores, indicar los dos apellidos de cada uno de los autores separados por "_". Por ejemplo, el estudiante "Perico Pirulo Palotes" utilizará el nombre de fichero: `PiruloPalotesPRA1.tgz`

Fecha de entrega

Entrega a través de Sakai el 2 de noviembre, entrega presencial en la clase de grupo medio durante la semana del 29 de octubre al 2 de noviembre de 2018.