Práctica 2. Concurrencia

Arquitectura y Tecnología de Computadores. Sistemas Distribuídos

1. Ejercicios a realizar

Los ejercicios a realizar consisten básicamente en los propuestos durante la sesión 2 de prácticas, con alguna variación. Se detallan a continuación.

1.1. Servidor para el protocolo *echo-reverso*, con concurrencia multi-proceso

Modificar el servidor de *echo-reverso* realizado en el trabajo anterior, de modo que admita hasta 4 clientes concurrentes, mediante la técnica de la creación previa de procesos usando `fork()`.

Cada proceso hijo realizará su trabajo en un bucle infinito y no tiene que morir después de atender a un cliente. Aún así, siempre será posible usar el comando `kill` para enviar a cualquiera de ellos una señal `SIGTERM` o `SIGKILL` que finalizaría su ejecución. Téngase esto en cuenta y prográmese de modo que al morir no quede *zombi*.

Téngase también en cuenta que, en algunos sistemas operativos es necesario garantizar la exclusión mutua entre estos procesos en la llamada a la función `accept()`. Aunque no sea el caso de las máquinas de prácticas, se pide que el código desarrollado implemente esta exclusión mutua. Para ello se sugiere usar la función `lockf()` cuyo funcionamiento se puede consultar en el manual de Unix (comando `man`), como se explicó en las clases prácticas.

1.2. Servidor para el protocolo *echo-reverso* con concurrencia aparente basada en `select()`

Modificar el servidor de *echo-reverso* realizado en el primer trabajo, de modo que admita hasta tres clientes con concurrencia aparente. Si un cuarto cliente intenta la conexión, el servidor la admitirá momentáneamente, pero inmediatamente después la cerrará sin darle ningún servicio.

El servidor deberá usar `select()` para no quedar bloqueado en las lecturas ni en el `accept()`. Además de atender eventos en el socket de escucha y de dar servicio en los sockets de datos, deberá atender eventos en la entrada estándar[1]. Cuando se detecta un evento en la entrada estándar, se lee un carácter mediante la función `getchar()` y el servidor deberá reaccionar ante las siguientes pulsaciones de tecla (ignorando cualquier otra):

's' (estadísticas) Muestra en la pantalla del servidor los siguientes datos:

- Número de clientes actualmente conectados.
- Número total de clientes aceptados desde que arrancó el servidor.
- Número total de clientes rechazados desde que arrancó el servidor.

'x' (expulsar) Cierra la conexión con todos los clientes que estuvieran conectados, y queda a la espera de nuevos clientes.

'q' (quit) Cierra la conexión con todos los clientes conectados y finaliza la ejecución del servidor.

[1] La entrada estándar, normalmente asociada al teclado, se trata en Unix como un fichero (o socket) más, cuyo descriptor entero tiene siempre el valor 0. Este valor entero también puede obtenerse evaluando la función `fileno()` sobre el objeto `stdin`.

2. Aspectos formales

Ambos servidores deberán admitir como parámetro en la línea de comandos el número de puerto en el que esperan clientes. Ambos funcionarán sobre protocolo TCP y por tanto podrán ser probados con `telnet`.

Debes crear una carpeta llamada `p2` y dentro de ella dos carpetas llamadas `e1` y `e2`, para cada uno de los ejercicios anteriores. Cada carpeta debe contener el código fuente (con los nombres de ficheros que detallaremos seguidamente) y alguna forma automatizada de compilación, ya sea en forma de *script* o de `Makefile`. El código fuente debe ser C estándar y debe compilar y funcionar al menos en las dos máquinas de prácticas `sirio` y `orion`.

Los nombres de los ficheros fuente serán los siguientes:

Ejercicio 1 `echo_reverso_fork.c`

Ejercicio 2 `echo_reverso_select.c`

2.1. Entrega

Cuando el código haya sido probado, se borrarán los ejecutables, dejando sólo el código fuente y el *script* o `Makefile` de compilación, y se empaquetarán todas las carpetas en un único archivo `tar` comprimido. Para esto, debes situarte en la carpeta "padre" de la carpeta `p2` y desde ella teclear:

```
$ tar czvf p2.tar.gz p2
```

El resultado será el fichero `p2.tar.gz`, que seguidamente deberás firmar digitalmente con el comando[2]:

```
$ gpg --sign p2.tar.gz
```

El resultado será un archivo `p2.tar.gz.pgp` que entregarás a través del formulario Web en la página de la asignatura. Si lo deseas, además de firmarlo puedes cifrarlo para tu profesor de prácticas, mediante el comando:

```
$ gpg --sign --encrypt -r jldiaz@uniovi.es -r ariasjr@uniovi.es p2.tar.gz
```

[2]Deberás efectuar esta operación en la máquina en la que hayas generado tu par de claves.