CC3301 Programación de Software de Sistemas
Tarea 4 – Semestre Otoño 2017 – Prof.: Luis Mateu

En esta tarea Ud. debe programar un servidor y un cliente para la cena de filósofos de la tarea 3. Por lo tanto esta es una continuación de esa tarea. El cliente sirve para que cada filósofo entregue su nombre y espere si es necesario hasta obtener una silla en donde sentarse. Luego el filósofo ingresa la palabra *fin* cuando terminó de comer. El servidor administra las sillas y se debe lanzar por medio del comando *restaurante*:

```
$ ./restaurante 3000
```

El único parámetro que recibe es el número del puerto que usa para ofrecer el servicio en el servidor.

El cliente se invoca mediante el comando *filosofo*. El único parámetro es el nombre del jugador. Este comando obtiene el nombre del servidor y el puerto en donde se ofrece el servicio a través de la variable de ambiente REST en el formato *host:puerto*, por ejemplo *anakena.dcc.uchile.cl:3000*.

La siguiente tabla muestra un ejemplo de uso de este sistema. Pedro, Juan, Diego y Luis son filósofos. Las filas están ordenadas cronológicamente. Se usan 5 terminales, cada uno ejecutando el shell de comandos. El directorio de trabajo inicial en cada terminal es el directorio en donde Ud. programó su tarea. Lo ingresado por el usuario aparece en **negritas**. El texto normal corresponde a lo desplegado por el programa. Además se incluyen las notas (A) (B) etc. cuya explicación aparece a continuación de este ejemplo.

shell 1: servidor	*shell 2: Pedro*	*shell 2: Juan*	*shell 2: Diego*	*shell 2: Luis*
$ **./restaurante 3000** (A)				
llega Pedro Pedro a silla 0	$ **REST=anakena:3000; export REST** (B) $ **./filosofo Pedro** (C) 0			
llega Juan Juan a silla 2		$ **REST=anakena:3000; export REST** $ **./filosofo Juan** (C) 2		
llega Diego			$ **REST=anakena:3000; export REST** $ **./filosofo Diego** (D)	
Juan libera 2 Diego a silla 2		**fin** (E) $	2	
llega Luis				$ **REST= … etc. …** $ **./filosofo Luis**
Pedro libera 0 Luis a silla 0	**fin** (E) $			0

Notas:

(A) Se lanza el servidor. Debe escuchar las solicitudes de conexión en el puerto 3000. Además despliega las llegadas de los filósofos, cuando se asigna una silla y cuando se libera.

(B) Se define la variable de ambiente REST con el nombre del servidor y el puerto en donde escucha.

(C) Llega un filósofo y obtiene una silla de inmediato. Ud. debe enviar el nombre del filósofo al servidor, luego recibir el número de la silla que se le otorgo y mostrarla en la salida estándar.

(D) Llega un filósofo pero no hay silla disponible así es que debe esperar.

(E) Un filósofo ingresa el texto *fin* seguido de un *newline* (tecla *enter*). Ud. debe leer ese texto de la entrada estándar. Luego enviar alguna indicación al servidor para señalar que el filósofo terminó de comer. Basta que envíe un solo carácter al servidor. En el servidor Ud. debe liberar la silla, lo que probablemente hará que un filósofo en espera pueda sentarse.

Requerimientos

- Programe los comandos *filosofo* y *restaurante* en los archivos *filosofo.c* y *restaurante.c*. Agregue en *restaurante.c* su solución de la tarea 3. Entregue su tarea solo si estos comandos reproducen exactamente la misma salida que se muestra en el ejemplo de este enunciado, exceptuando la notas (superíndices en letra cursiva y entre paréntesis).
- El comando *restaurante* se termina ingresando *control-C* en el shell en donde se ejecuta.
- En el servidor Ud. debe usar threads para conversar con los clientes.
- Use la función *getenv* en *filosofo.c* para obtener el valor de la variable de ambiente REST. Obtenga su documentación con *man getenv*.
- Ud. debe usar en el servidor su solución de la tarea 3, aún si esta no logró pasar los tests de robustez sobre detección de dataraces. La probabilidad de que un datarace se manifieste en el ejemplo en la tarea 4 es 0. Si Ud. no entregó a tiempo la tarea 3, igual deberá resolverla personalmente para poder entregar la tarea 4. No use la solución de un compañero de curso.

Recomendaciones

- Resuelva esta tarea antes del control 3. Le servirá de estudio.
- Si prueba su tarea en *anakena*, use un puerto distinto de 3000 puesto que sus compañeros también estarán probando la tarea en *anakena*. Recuerde que 2 procesos no pueden usar el mismo puerto de la misma máquina.

Recursos

- Baje *t4.zip* del material docente de U-cursos y descomprímalo. El directorio contiene el archivo *Makefile* para compilar su tarea y los archivos para usar sockets (*libjsocket.c* y *util.c*).

Entrega

Ud. debe entregar un archivo *".zip"* con los archivos *restaurante.c* y *filosofo.c* por medio de U-cursos. No incluya otros archivos. Sobretodo no incluya archivos binarios. Se descontará medio punto por día de atraso. No se consideran los días sábado, domingo o festivos.