CC3301 Programación de Software de Sistemas
Tarea 4 – Semestre Primavera 2016 – Prof.: Luis Mateu

En esta tarea Ud. debe programar un servidor y un cliente para organizar los equipos de baby-futbol de la tarea 3. Por lo tanto esta es una continuación de esa tarea. El cliente sirve para que cada jugador entregue su nombre y espera si es necesario hasta que el equipo esté formado. Luego le entrega al jugador los nombres de los 5 jugadores del equipo al que pertenece y termina. El servidor se debe lanzar por medio del comando *organizador*:

```
$ ./organizador 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 *hay-equipo*. 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 ORGSVR 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, Jaime y Luis son jugadores. Las filas están ordenadas cronológicamente. Se usan 10 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) y (B) cuya explicación aparece a continuación de este ejemplo.

shell 1: Pedro	***shell 2: Juan***	***...***	***shell 5: Diego***	***shell 6: Jaime***	***...***	***shell 10: Luis***
$ **ORGSVR=anakena:3000; export ORGSVR** $ **./hay-equipo pedro** *(A)*						
	$ **ORGSVR=anakena:3000; export ORGSVR** $ **./hay-equipo juan** *(A)*					
	… *(B)*					
pedro juan … diego $	pedro juan … diego $		$ **ORGSVR=anakena:3000; export ORGSVR** $ **./hay-equipo diego** *(C)* pedro juan … diego $			
				$ **ORGSVR=anakena:3000; export ORGSVR** $ **./hay-equipo jaime** *(A)*		
				… *(B)*		
				jaime … luis $		$ **ORGSVR=anakena:3000** $ **export ORGSVR** $ **./hay-equipo luis** *(C)* jaime … luis $

Notas:

(A) Un jugador usa el comando *hay-equipo* para indicar su nombre. Este comando se conecta al servidor *organizador* en anakena que escucha en el puerto 3000. El comando espera hasta que se forme el equipo.

(B) Además de los 5 jugadores que aparecen explícitamente en la tabla, hay otros 5 jugadores que usan el comando *hay-equipo* para completar 2 equipos de baby-futbol en total. Su interacción queda implícita en filas etiquetadas con …

(C) Este jugador es el último que se necesita para formar el equipo. Todos los comandos *hay-equipo* pendientes terminan entregando los nombres de los jugadores pertenecientes al equipo.

Requerimientos

- Programe los comandos *hay-equipo* y *organizador* de este sistema. Entregue su tarea solo si el comando *hay-equipo* reproduce exactamente la misma salida que se muestra en el ejemplo de este enunciado, exceptuando la notas (superíndices en letra cursiva y entre paréntesis).
- Cuando llega el último jugador, todos los comandos *hay-equipo* pendientes deben terminar.
- El comando *organizador* 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* para obtener el valor de la variable de ambiente ORGSVR. Obtenga su documentación con *man getenv*.
- Ud. debe usar en el servidor su solución de la tarea 3, aún si esta no logró pasar el test 2 sobre detección de dataraces. La probabilidad 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.
- Cuando pruebe 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*).
- Programe el servidor en el archivo *organizador.c* y el cliente en *hay-equipo.c*.

Entrega

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