PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE
ESCUELA DE INGENIERÍA
DEPARTAMENTO DE CIENCIA DE LA COMPUTACIÓN

IIC2333 — Sistemas Operativos y Redes — 2/2017

Tarea 2

Lunes 4-Septiembre-2017

Fecha de Entrega: Viernes 15-Septiembre-2017 a las 23:59
Composición: grupos de n personas, donde $n \leq 2$

Esta tarea se compone de dos partes, independientes entre si, donde deberán utilizar las *syscalls* de manejo de procesos, y la biblioteca POSIX Threads.

Parte I. myShell - `msh`

El objetivo de esta parte es construir un intérprete de comandos básico que llamaremos `msh`. Deberá recibir por `stdin` la ubicación de un ejecutable, junto con una cantidad **arbitraria** de argumentos. El ejecutable debe ser invocado con los parámetros dados, y cuando éste termine se deberá poder especificar otro ejecutable y otros parámetros para repetir el proceso. Abajo se puede ver un ejemplo de funcionamiento.

```
> Ingrese un comando: /bin/echo Hello World
Hello World
> Ingrese un comando: /bin/date
Mon Sep  4 19:30:21 CLST 2017
> Ingrese un comando: /usr/bin/time /bin/echo Hello World
Hello World
0.00user 0.00system 0:00.00elapsed 0\%CPU (0avgtext+0avgdata 1516maxresident)k
0inputs+0outputs (0major+68minor)pagefaults 0swaps
> Ingrese un comando:
```

`msh` debe reconocer algunos comandos propios (*built-in*):

- `exit` provoca que `msh` termine inmediatamente.
- `setPrompt string` provoca que el texto que precede al comando que ingresará el usuario sea `string`. Puede definir un *prompt* por defecto. Si `string` contiene un `*`, este se reemplaza por el *exitCode* del último proceso ejecutado.
- `setPath path` provoca que los ejecutables ubicados en `path` se puedan ejecutar sin necesidad de ingresar la ruta absoluta completa. Por ejemplo: si se ejecutó `setPath /bin/`, entonces al escribir `ls` se ejecuta lo equivalente a escribir `/bin/ls`.
- Si el último argumento es `&`, el proceso debe ser ejecutado en el *background*, esto significa que el comando continuará ejecutando, pero `msh` podrá recibir otro comando.
- Si el último argumento es `&N`, el proceso debe ser ejecutado `N` veces en **paralelo** (esto significa que pueden estar en estado *ready* simultánamente).

Al presionar Ctrl C el comportamiento varía según si se está ejecutando un comando o no. Si se está ejecutando un comando, debe terminarse el comando (y no la shell básica). En caso contrario, debe terminar `msh`.

Tenga en cuenta

1. Deberá usar `fork`, `exec` y `wait` (no *threads*).
2. Deberá construir `argc` y `argv` para pasarlos al comando que se va a ejecutar.

Parte II. Juego de la vida - `life`

El juego de la vida[1] es una simulación del tipo autómata celular, cuya evolución sólo depende del estado anterior. Consiste en una matriz de $n \times m$ celdas, donde cada celda tiene uno de dos estados posibles: viva o muerta. Para pasar de un estado a otro existen las siguientes reglas:

1. Una celda en (i, j) es vecina de otra celda en $(i', j') \Leftrightarrow$ son distintas y $|i - i'| \leq 1 \wedge |j - j'| \leq 1$.
2. Toda celda viva con menos de dos vecinos vivos, muere.
3. Toda celda viva con más de tres vecinos vivos, muere.
4. Toda celda muerta con **exactamente** tres vecinos vivos, se convierte en una celda viva.

El objetivo de esta parte es construir un simulador de este autómata utilizando procesos, *threads* y memoria compartida, en un modelo de *master* con *workers*.

Su programa deberá aceptar un *input* con el siguiente formato mediante `stdin`:

1. La primera línea contendrá cinco números enteros t, f, c, v, n separados por un espacio entre ellos, donde t es es el número de iteraciones a simular, f el número de filas de la matriz, c el número de columnas, v el número de celdas vivas en el estado inicial, y n el número de *threads* que usará cada *worker*. Siempre se tendrá que $t, f, c, v, n > 0$ y $0 < v \leq f \times c$.
2. Cada una de las siguientes v líneas contendrá dos números enteros i, j que representan la posición de una celda viva en el estado inicial, donde i corresponde al número de la fila, y j al número de la columna. Siempre se tendrá que $0 \leq i < f$ y $0 \leq j < c$, y también puede asumir que cada celda no aparece más de una vez.

El programa debe generar como *output* f líneas, donde cada línea representa una fila de la matriz después de t iteraciones. Cada línea tendrá c números enteros, que es 0 si en esa posición hay una celda muerta, o 1 si hay una celda viva. Abajo se muestra un ejemplo de input y output esperado [2].

Input de ejemplo

```
1 3 3 3
0 1
1 1
2 1
```

Output de ejemplo

```
0 0 0
1 1 1
0 0 0
```

Tenga en cuenta

1. El proceso *master* debe generar tantos **procesos** de tipo *worker* como **cores** tenga la CPU del sistema en que se ejecuta el programa. Cada *worker* deberá simular las t iteraciones y, en cada iteración, reportar al proceso *master* el número de iteraciones que ha realizado y su estado actual. Cuando uno de los *workers* llegue al estado final de la simulación, deberá terminar (*kill*) a los demás *workers* y entregar el *output* que obtuvo.
2. Los procesos *master* y *workers* (deben ser procesos, no *threads*) deberán trabajar sobre un espacio de memoria en común. Esto aplica para el reporte que hacen los procesos *workers* al proceso *master*.

[1] Juego de la Vida en *Wikipedia*
[2] Demo Juego de la Vida

3. Cada *worker* trabaja sobre una copia de la matriz inicial. Para esto, cada *worker* deberá generar n *threads*, entre los cuales se divide equitativamente el trabajo de actualizar la matriz sobre la cual trabaja dicho *worker*.
4. Sólo debe crear los *workers* una vez.
5. Es muy probable que requiera coordinar los distintos *threads*. **Si no utiliza ninguna forma de coordinarlos, estará obligado a especificar por qué no fue necesario en su `README`.**

README y Formalidades

Deberá incluir un archivo `README` que indique quiénes son los autores de la tarea (**con sus respectivos números de alumno**), cuáles fueron las principales decisiones de diseño para construir el programa, qué supuestos adicionales ocuparon, y cualquier información que considere necesarias para facilitar la corrección de su tarea. Se sugiere utilizar formato **markdown**.

A cada alumno se le asignó un nombre de usuario y una contraseña para el servidor del curso (`iic2333.ing.puc.cl`). Para entregar su tarea usted deberá crear una carpeta llamada `T2` en el directorio `Entregas` de su carpeta personal y subir su tarea a esa carpeta. En su carpeta `T2` **solo debe incluir código fuente** necesario para compilar su tarea, además del README y un `Makefile`. **NO debe incluir archivos binarios (será penalizado)**. Se revisará el contenido de dicha carpeta el día Viernes 15-Septiembre-2017 a las 23:59.

- Puede ser realizada en forma individual, o en grupos de 2 personas. En cualquier caso, recuerde indicar en el `README` los autores de la tarea con sus respectivos números de alumno.
- La parte I de esta tarea debe estar en el directorio `Entregas/T2/msh`, y la parte II de esta tarea en el directorio `Entregas/T2/life`. Cada una de las partes debe compilar utilizando el comando `make` en los directorios señalados anteriormente.

Evaluación

- 0.5 pts. Formalidades. Esto incluye cumplir las normas de la sección formalidades.
- 2.5 pts. Shell básica
  - 1 pts. Lectura de `stdin`. Paso de argumentos. Construcción de `argc` y `argv`.
  - 1.5 pts. Funcionamiento correcto utilizando *syscall*s de procesos.
- 3 pts. Juego de la vida.
  - 1 pts. Correctitud del algoritmo, input y output.
  - 2 pts. Uso de memoria compartida, procesos, *threads* y sincronización entre ellos.

El no respeto de las formalidades o un código extremadamente desordenado podría originar descuentos adicionales. Se recomienda modularizar, utilizar funciones y ocupar nombres de variables explicativos. En el caso de no entregar en la carpeta especificada la tarea **no** se corregirá.

Bonus (+0.5 pts): manejo de memoria perfecto

Se aplicará este bonus si *valgrind* reporta en su código 0 *leaks* y 0 errores de memoria, considerando que los programas funcionen correctamente. El bonus a su nota se aplica solo si la nota correspondiente es $\geq 3{,}95$.

Preguntas

Cualquier duda preguntar a través del foro.