Universidad Simón Bolívar
Departamento de Computación y Tecnología de la Información
CI-3825 - Sistemas de Operación I
Trimestre Septiembre-Diciembre de 2016

Proyecto I: Campo de Batalla (15 %)

Objetivos Generales

- Familiarizarse con los mecanismos de creación y manejo de procesos e hilos de ejecución en GNU/Linux.
- Familiarizarse con mecanismos básicos de comunicación entre procesos, y entre hilos

Definición del problema

En un país lejano pero muy lejano, el Comando Militar (CM) está planeando cómo destruir un campo de batalla enemigo. Sin embargo, en el mismo campo se encuentra áreas pobladas, por lo que deben ser muy cuidadoso. En un *escenario simulado*, el campo de batalla se describe como una área cuadrada de $N \times N$ de yardas. Las yardas están numeradas de 0 a $N-1$. En este *escenario simulado*, los objetivos militares (OM) se marcan con números enteros negativos, donde el número describe la "resistencia" del objetivo. Se tiene que un (OM) es más resistente, mientras menor sea el valor asignado. Por lo que, un objetivo -8 es más resistente que uno de -2. Los objetivos civiles (OC) se marcan de la misma manera, pero utilizando números enteros positivos, que indican la importancia del objetivo. Mientras mayor sea el número asignado a OC, mayor es su importancia. Por lo que, un OC de 8 es más importante que un OC de 2. Las yardas sin interés se marcan con el número 0.

Los expertos del CM programan una serie de ataques con bombas especiales que pueden destruir áreas cuadradas del *escenario simulado*. Cada una de tales bombas, tiene cuatro parámetros: (I) la coordenada en el eje X de la posición donde va a caer la bomba, tal que $0 \leq X \leq N-1$. (II) la coordenada en el eje Y de la posición donde va a caer la bomba, tal que $0 \leq Y \leq N-1$ (III) su radio de alcance R y (IV) la potencia de la misma P. Todos los parámetros son números enteros. Cuando una bomba es lanzada en el *escenario simulado* todas las yardas comprendidas en el cuadrado con coordenadas, $(X+R,\, Y+R)$, $(X+R,\, Y-R)$, $(X-R,\, Y+R)$ y $(X-R,\, Y-R)$ son afectadas por la bomba. Las bombas serán efectivas sólo en el campo de batalla, por lo que coordenada menor que 0 o mayor que $N-1$ no debe ser consideradas. En consecuencia, cualquier objetivo OM o OC que se encuentre dentro del área de destrucción

de la bomba se ve afectado. Si una bomba alcanza a un OM, su efecto sobre la yarda del OM es el de sumar a su resistencia la potencia P de la bomba. Cuando el objeto alcanzado por una bomba es un OC, se le resta P unidades a la importancia del OC. El efecto de las bombas sobre los objetivos OCs y OMs es acumulativo.

Después de una serie de ataques con bombas, CM quiere saber:

- ¿Cuántos OM no fueron alcanzados?
- ¿Cuántos OM fueron parcialmente destruidos?
- ¿Cuántos OM fueron totalmente destruidos?
- ¿Cuántos OC no fueron alcanzados?
- ¿Cuántos OC fueron parcialmente destruidos?
- ¿Cuantas OC fueron totalmente destruidos?

Su misión es la de ayudar al MC determinar dicha información. Para ello debe de escribir un programa que contiene dos implementaciones que solucione el problema. La primera resuelve el problema haciendo uso de *procesos*. La segunda implementación utiliza *hilos* o *threads*.

Ejecución del programa

A partir de sus códigos fuentes se debe generar un archivo ejecutable llamado `batalla`. El comando para su ejecución es como sigue:

```
>./batalla [-p] [-h] [-n <cantidad>] <archivo_entrada>
```

donde:

`-p`: indica que el programa usa procesos para resolver el problema.
`-h`: indica que el programa usa hilos(threads) para resolver el problema.
`-n <cantidad>`: establece el número de procesos o hilos que ayudan al proceso principal. En `<cantidad>` se indica cuántos.
`<archivo_entrada>`: es un archivo con el campo de batalla y el grupo de bombas para el bombardeo.

Las opciones `-n`, `-h` y `-p` son optativas. El valor por defecto de `-n` es 0, es decir, el problema es resuelto solamente por el procesos padre. La opción de ejecución por defecto es `-p`. El parámetro `archivo_entrada` es obligatoria. Las opciones `-p` y `-h` son excluyentes, en caso de presentarse juntas se debe dar un mensaje de error al usuario.

Suponiendo que tenemos un archivo con el formato adecuado llamado `bombardeo1.txt`, a continuación de muestran 4 ejemplos de llamadas válidas al programa `batalla`:

```
>./batalla bombardeo1.txt
>./batalla bombardeo1.txt -n 2 -p
```

```
>./batalla -h  bombardeo1.txt -n 4
>./batalla -p -n 3 bombardeo1.txt
```

Entrada de los datos

Los valores del *escenarios simulados* son leídos desde un archivo. El primer número en el archivo describe el tamaño N del campo de batalla. En una segunda línea, el número T de objetivos, incluyendo tanto OC como OM. A continuación, T líneas con tres números enteros separados por un espacio en blanco: las coordenadas X y Y del objetivo y el valor asociado del objetivo el cual será positivo para un OC, negativo para un OM. Después de las T líneas, se presenta un número B que indica la cantidad de ataques, o bombas a lanzar, planeadas por el CM. Las siguientes B líneas describen cada uno de los ataques. Un ataque se describe mediante cuatro cifras separados por un espacio en blanco: las coordenadas X y Y, el radio de alcance R, y la potencia P de la bomba. El número de ataques B puede ser cualquiera, y las coordenadas X y Y se podrían repetir entre los ataques . Se garantiza que hay al menos un OC y un OM, y que no hay dos objetivos (OC y/o OM) con las mismas coordenadas.

Salida de los datos

Después analizar los B ataques, se debe escribir en la salida estándar el resultado en el siguiente formato:

Objetivos Militares totalmente destruidos: A
Objetivos Militares parcialmente destruidos: B
Objetivos Militares no afectados: C
Objetivos Civiles totalmente destruidos: D
Objetivos Civiles parcialmente destruidos: E
Objetivos Civiles no afectados: F

Donde A, B, C, D, E y F son las cantidades respectivas. La salida del programa debe hacerse por la salida estándar.

Ejemplo práctico

Sea un archivo llamado `ataque1.txt` un archivo con un campo de batalla y datos de bombardeo con el siguiente contenido:

```
10
4
0 0 8
5 5 100
1 1 -2
```

```
7 7 -6
5
2 1 2 3
1 1 1 4
7 7 0 3
6 6 4 8
9 9 8 1
```

Si se ejecuta:

```
./batalla -n 5 ataque1.txt
```

El programa `batalla` resuelve el problema con un proceso padre que tiene la ayuda de 5 procesos hijos, obteniéndose por la salida estándar el siguiente resultado:

```
Objetivos Militar  totalmente destruidos: 2
Objetivos Militar  parcialmente destruidos: 0
Objetivos Militar  no afectados: 0
Objetivos Civil  totalmente destruidos: 0
Objetivos Civil parcialmente destruidos:  2
Objetivos Civil  no afectados: 0
```

Requerimientos de la implementación

El programa se inicia en el proceso principal quien recibe el número de ayudantes, procesos o hilos, que deben cooperar con él y el nombre del archivo de entrada, el cual contiene la descripción del campo de batalla y el detalle del bombardeo. El proceso principal debe construir el campo de batalla. En pasos siguientes el proceso principal debe *distribuir las bombas entre los procesos ayudantes*. Si el número de procesos ayudantes es mayor que el número de bombas, entonces el programa sólo opera con un número de procesos ayudantes igual al número de bombas. Cada proceso ayudante tiene como una de sus funciones, el procesamiento de la(s) bomba(s) sobre el campo de batalla. Una vez procesadas todas las bombas, cada proceso ayudante debe informar al proceso principal el resultado obtenido. El proceso principal se encargará de calcular el efecto total sobre el campo y de escribir el resultado por la salida estándar.

Recomendaciones

1. Se debe chequear el valor de retorno de las llamadas al sistema.

2. Deben hacer un programa modular, legible y documentado.
3. Los programas deben poder compilarse y ejecutarse en cualquiera de las computadoras (GNU/Linux) del LDC. Si Ud. realizó el programa en su casa o en la USB, pero en alguna otra plataforma, debe asegurarse antes de la entrega que su proyecto funciona en las estaciones GNU/Linux antes mencionadas.
4. Para el procesamiento de los argumentos de la línea de comandos se recomienda usar `getopt` o `argp_parse`, ver: `https://goo.gl/WrB8E7`
5. Se debe asegurar que su programa no tiene pérdidas de memoria. Para ello se sugiere el uso de **Valgrind** [1].

Informe del proyecto

Debe realizar un informe, en formato PDF, con la siguiente estructura:

1. **Portada**
2. **Diseño de la solución:** explicación de las estructuras datos y algoritmos usados en la solución del problema. Justificación de cada uno de ellos.
3. **Detalles de implementación:** reseña y explicación de los elementos implementados.
4. **Estudio experimental y discusión:** El objeto del estudio es observar la diferencias de rendimiento entre la implementación hecha con procesos y la realizada con hilos. También se quiere que muestre como el rendimiento cambia a medida que se aumenta el número de procesos/hilos que son usados en la resolución del problema. Para un problema no trivial, debe realizar al menos 6 pruebas de su programa con diferentes números de procesos e hilos que van desde de 0 hasta N_b, donde N_b es el número de bombas del problema a resolver. Asegúrese de hacer pruebas con un número de procesos/hilos que sea menor que el número procesadores o núcleos del computador en donde se realicen los experimentos. Una vez obtenido los resultados, debe hacer un análisis de los mismos. En el informe también debe indicar los detalles de la plataforma usada para realizar los experimentos, esto es, (I) el sistema de operación, (II) la versión del compilador, (III) el modelo del CPU de la computadora usada incluyendo los datos de la velocidad de reloj, el número de procesadores físicos, el número de *cores* de cada procesador y el número *threads* que se pueden ejecutar simultáneamente en cada *core*. (IV) la cantidad de memoria RAM de la computadora usada.
5. **Estado actual:** estado del programa entregado, es decir, si está totalmente operativo, parcialmente operativo, etc. Indicar errores observados en sus pruebas.
6. **Referencias bibliográficas:** en caso de hacer uso de alguna.

[1] `http://valgrind.org/`

Condiciones de la entrega

La entrega del proyecto será el día **28 de Octubre de 2016**. La entrega tiene las siguientes condiciones:

1. Debe crear un llamado `Proyecto1-X-Y.tar.xz`, donde `X` y `Y` son los número de carné de los integrantes del grupo, con todos los códigos fuentes (archivos .c y .h), el Makefile y el informe, el cual debe ser entregado en la página del curso en el Aula Virtual, el día viernes antes de las 1:00 pm.
2. Proyecto que no pueda compilarse usando las reglas dadas en el archivo Makefile, tiene CERO como nota.
3. El programa debe hacer uso eficiente de los recursos que administra el sistema operativo: memoria, tiempo, etc.
4. Debe respetar las especificaciones que se le dan en el enunciado.
5. Debe entregar a la hora del taller, la **Declaración de autenticidad de entrega**, debidamente firmada por los integrantes del equipo.