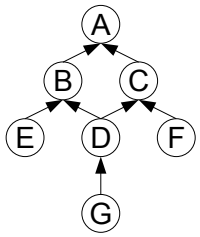
CC41B Sistemas Operativos

Tarea 1 – Semestre Primavera 2008

Prof.: Luis Mateu



El procedimiento **`seqMake`** sirve para compilar un programa respetando el orden dado por un grafo de dependencias. A modo de ejemplo considere el grafo de la figura. El grafo muestra que el archivo A se genera a partir de B y C, B se genera a partir de E y D, etc. Observe que tanto B como C dependen de D, el cual se genera compilando G. Los archivos E, G y F son preexistentes y por lo tanto no se generan a partir de otros archivos. En general las dependencias forman un *grafo dirigido acíclico*.

La siguiente es una implementación secuencial de **`seqMake`**:

```
typedef struct SrcFile {
  char *name;
  struct SrcFile **deps; /* arreglo de dependencias */
} SrcFile;

void seqMake(SrcFile *file) { seqMakeRec(makeSet(), file); }

void seqMakeRec(Set *readySet, SrcFile *file) {
  if (!contains(readySet, file->name)) {
    int i;                                /* No ha sido compilado aun. */
    for (i= 0; file->deps[i]!=NULL; i++)  /* Primero compilar  */
      seqMakeRec(readySet, file->deps[i]); /* las dependencias. */
    compile(file);                        /* Compilar y agregar a */
    add(readySet, file->name);            /* readySet para que no se   */
  }                                       /* recompile.                */
}
```

`seqMake` recibe como parámetro el grafo de dependencias y retorna cuando cada nombre de archivo en el grafo fue generado respetando el grafo de dependencias. El procedimiento `compile(file)` genera el archivo `file->name` a partir de sus dependencias `file->deps` suponiendo que estos archivos fueron generados previamente (invocando `compile`) o ya existen (no dependen de otros archivos). La única operación que requiere mucho tiempo de CPU es `compile`.

Requerimientos

Programe el procedimiento void **`parMake(SrcFile *file, int p)`** correspondiente a una versión paralela de **`seqMake`**. **parMake** debe implementar la misma funcionalidad que **seqMake**. En **parMake** Ud. debe hacer invocaciones paralelas de `compile`, permitiéndose hasta un máximo de **p** llamadas simultáneas. Debe respetar las dependencias de la misma forma que lo hace **`seqMake`**: no debe iniciar la generación de A antes que termine la generación de B, puesto que B se usa para generar A. No esta permido generar un mismo archivo más de una vez y por lo tanto debe tener cuidado en no generar 2 veces D. Para resolver este problema Ud. debe usar las tareas y monitores de nSystem. El tipo `Set` no admite llamadas concurrentes.

Archivos suministrados

Se provee (i) el archivo `testmake.c` con el programa de prueba en donde se implementan los procedimientos `nMain` y `compile`, (ii) el archivo `srcfile.h` con la definición de la estructura `SrcFile` y el encabezado de `parMake`, (iii) los archivos `set.h` y `set.c` en donde se definen e implementan los procedimientos `makeSet`, `contains`, `add` y `destroySet`, (iv) el archivo `Makefile` para compilar su tarea (secuencialmente), y (v) la implementación secuencial **`seqMake`** en el archivo `make.c.template`.

Entrega

Ud. debe programar, verificar y entregar el archivo `make.c` en donde implementa el procedimiento **`parMake`**. Pronto se indicará como verificar que su implementación funciona de acuerdo a los requerimientos. Entregue el archivo `make.c` en U-cursos. No se aceptarán tareas que no funcionen. El plazo de entrega vence el Lunes 8 de Septiembre. Se descontará medio punto por dia hábil de atraso. Se recomienda vivamente resolver esta tarea antes del control 1.