En esta tarea Ud. deberá programar el comando *compilar* que recibe como parámetro el nombre de un directorio *dir* y lo recorre en profundidad mostrando en su salida estándar todos los archivos con extensión *.c* que necesitan compilarse ordenados alfabéticamente. El siguiente es un ejemplo de uso:

```
$ ./compilar.bin dir
dir/222/ejemplo-sort.c
dir/sort-c.c
dir/test-comprimir.c
dir/test-elim-rango.c
dir/test-reemplazar.c
```

Los archivos que necesitan compilarse son aquellos para los cuales ***(i)*** no existe un archivo con el mismo nombre con extensión *.o* en lugar de *.c* como por ejemplo *dir/222/pss.o*, o ***(ii)*** sí existe el archivo con el mismo nombre con extensión *.o*, pero la fecha del *.o* es anterior a la fecha del *.c*. La fecha está en el campo *st_mtime* de la estructura obtenida con *stat* (en segundos).

Metodología obligatoria

Para recorrer en profundidad el directorio, base su solución en el programa *list-dir.c* que viene en los archivos adjuntos y que lista recursivamente los archivos y directorios a partir del parámetro recibido. Se compila e invoca con:

```
$ make PROB=list-dir list-dir.bin
$ ./list-dir.bin dir
dir
dir/sort-c.c
dir/test-elim-rango.c
...
```

Para ordenar los resultados le serán de mucha utilidad la cola *Queue* y la función *sortPtrArray* programadas en *pss.c*. Estudie los encabezados de las funciones en *pss.h*. Hay un ejemplo de uso de *sortPtrArray* en *ejemplo-sort.c*. Se compila y ejecuta con:

```
$ make PROB=ejemplo-sort ejemplo-sort.bin
$ ./ejemplo-sort.bin
…
```

Declare una cola global. Al recorrer recursivamente el directorio, por cada archivo *.c* que necesita compilarse agregue su nombre a la cola. Al terminar calcule el tamaño de la cola con la función *queueLength*. Esa es la cantidad de nombres que necesita ordenar. Transfiera los nombres en la cola a un arreglo del mismo tamaño que la cola y ordene el arreglo con la función *sortPtrArray*. Deberá definir una función para establecer que el criterio de ordenamiento es el orden alfabético, el decir el mismo de *strcmp*. Finalmente recorra el arreglo mostrando los nombres en la salida estándar.

Cuidado: el programa *list-dir.c* pide memoria con *malloc* para los nombres de los archivos encontrados y después libera esa memoria. Si agrega esos mismos strings a la cola, al finalizar será referencias colgantes. Deposite en la cola una copia de esos strings obtenida mediante la función *strdup*. No se preocupe por la eficiencia, pues no se exige en esta tarea. Recuerde que Ud. debe liberar la memoria que *sanitize* diagnostique como gotera de memoria.

Instrucciones

Descargue *t7.zip* de U-cursos y descomprímalo. Ejecute el comando *make* sin parámetros en el directorio *T7* para recibir instrucciones acerca del archivo en donde debe programar su solución (*T7/compilar.c*), cómo compilar y probar su solución, los requisitos que debe cumplir para aprobar la tarea (*make run-san*, *make run-g* y *make run*) y cómo entregar su tarea por U-cursos (*make zip*).

Entrega

Ud. solo debe entregar por medio de U-cursos el archivo *compilar.zip* generado por el comando *make zip*. **A continuación es muy importante que descargue de U-cursos el mismo archivo que subió, luego descargue nuevamente los archivos adjuntos y vuelva a probar la tarea tal cual como la entregó**. Esto es para evitar que Ud. reciba un 1.0 en su tarea porque entregó los archivos equivocados. Créame, sucede a menudo por ahorrarse esta verificación. Se descontará medio punto por día de atraso. No se consideran los días de receso, sábados, domingos o festivos.