En esta tarea Ud. deberá programar un comando que determine si una jerarquía de archivos y directorios es subconjunto de otra. Se invoca por ejemplo con:

```
$ ./cmp-dir.bin dir1/igualesB dir2/igualesB
Es subconjunto
```

El comando recibe 2 parámetros A y B que pueden ser archivos regulares o directorios. Que A sea subconjunto de B significa que: (i) Si A es un archivo regular (normal), entonces B también es un archivo regular y tiene el mismo tamaño y contenido; (ii) Si A es un directorio, entonces B también es un directorio y para toda entrada X de A (es decir un archivo de la forma A/X) debe existir también una entrada en B con el mismo nombre X, y además A/X es subconjunto de B/X. Note que el directorio B puede contender entradas que no están en A. Si no se cumplen estas condiciones, A no es subconjunto de B. El comando termina cuando se encuentra la primera inconsistencia, explicando la diferencia.

Los siguientes son más ejemplos de la salida del programa con los datos de prueba suministrados. Ud. debe entregar exactamente los mismos mensajes.

```
$ ./cmp-dir.bin dir1/contenidos-distintos \
                dir2/contenidos-distintos
Contenidos de dir1/contenidos-distintos/cardinales.txt y
dir2/contenidos-distintos/cardinales.txt difieren
$ ./cmp-dir.bin dir1/tamanos-distintos \
                dir2/tamanos-distintos
dir1/tamanos-distintos/cardinales.txt y dir2/tamanos-
distintos/cardinales.txt son de distinto tamanno
$ ./cmp-dir.bin dir1/noexiste2 dir2/noexiste2
dir1/noexiste2/hola.txt si existe,
dir2/noexiste2/hola.txt no existe
$ ./cmp-dir.bin dir1/noexiste1 dir2/noexiste1
dir1/noexiste1 no existe
$ ./cmp-dir.bin dir1/dir-inconsistente \
                  dir2/dir-inconsistente
dir1/dir-inconsistente es directorio, dir2/dir-
inconsistente no es directorio
$ ./cmp-dir.bin dir1/dir-inconsistente2 \
                  dir2/dir-inconsistente2
dir1/dir-inconsistente2 no es directorio, dir2/dir-
inconsistente2 si es directorio
```

En algunos de los ejemplos, la salida del comando aparece cortada en 2 líneas porque no cabe, pero es solo una línea.

Además Ud. debe diagnosticar los siguientes errores usando la función *perror* y terminar el programa con un código de retorno 1:

```
$ ./cmp-dir.bin dir1/error-arch dir2/error-arch
dir2/error-arch/sin-permiso: Permission denied
$ echo $?
1
$ ./cmp-dir.bin dir1/error-dir dir2/error-dir
dir1/error-dir: Permission denied
$ echo $?
1
```

Se permite que el mensaje de error esté en el idioma de la instalación de su computador.

Para recorrer el primer directorio (*no necesita recorrer el segundo directorio*), base su solución en el programa *listdir.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 list-dir.bin ; ./list-dir.bin dir1
dir1
dir1/iguales
dir1/iguales/cardinales.txt
dir1/dir-inconsistente
…
```

Restricciones:

- Debe usar las funciones *opendir*, *readir* y *closedir* para leer los directorios.
- Debe usar las funciones *open*, *read* y *close* para leer los archivos y compare contenidos con *memcmp*. Está prohibido usar *fopen*, *fread* y *fclose*.

Instrucciones

Baje *t6.zip* de U-cursos y descomprímalo. Ejecute el comando *make* sin parámetros. Le explicará cómo compilar y ejecutar, y los requisitos que debe cumplir su tarea para ser aprobada.

Entrega

Ud. debe entregar por U-cursos el archivo *cmp-dir.zip* que genera el comando *make zip*. **Recuerde descargar el archivo que subió, descomprimirlo e inspeccionar el contenido para verificar que son los archivos correctos.** Se descuenta medio punto por día de atraso, sin considerar recesos, sábados, domingos y festivos.