CMPUT 379, Assignment 1, Winter 2018

(UNIX signals, file system operations)

Objective

You are asked to write a C program that periodically reports the changes to the contents of a directory. The program should be able to, in principle, run forever, unless the process is terminated (see also the remaining specifications). In addition to the periodic reporting on the monitored directory, a report can also be triggered by sending the process a `SIGUSR1` Unix signal.

The program is invoked as follows:

`diffdir379` *period path*

Where *path* is the absolute or relative path to the directory to monitor, and *period* is the reporting period in seconds.

The output obeys the following format:

```
...
Sun Nov 5 13:10:50 MST 2017
- queue.c
+ newqueue.c
* driver.c
* driver.h
d Runs
...
```

That is, the date string in one line, followed by as many lines as the files/directories/others that have changed (could be zero). Each line for a changed file indicates, using a single character at the beginning of the line, what is the kind of object that gets created (`+` for a regular file, `d` for a directory, `o` for anything else, e.g., symbolic links, named pipes, etc. -- you will familiarize yourselves with `readdir()` to understand this part). If something is already created, then the character in the beginning indicates whether it has been modified (*), or was removed (`-`). Notice that the directories are not to be traversed, i.e., you only report about the objects within the specified directory. In the above example, a new regular file was created (`newqueue.c`) since last time, a file was removed (`queue.c`), two files were modified (`driver.c` and `driver.h`), and a new directory (`Runs`) was created. Notice that there exists exactly a single space character between the single-character indicator and the file name. There should be no empty lines between periodic reports, for example, the following (every 5 seconds) report:

```
...
Sun Nov 5 13:10:40 MST 2017
Sun Nov 5 13:10:45 MST 2017
Sun Nov 5 13:10:50 MST 2017
- queue.c
+ newqueue.c
* driver.c
* driver.h
d Runs
Sun Nov 5 13:10:55 MST 2017
o oldruns
Sun Nov 5 13:11:00 MST 2017
Sun Nov 5 13:11:05 MST 2017
...
```

indicates that no changes happened between :10:40 and :10:45, :10:55 and :11:00, etc. and the change between :10:50 and :10:55 was a creation of an "other" type of object (e.g., symbolic link).

In addition to a periodic report, a report is produced every time the process receives a SIGUSR1 signal. These "forced" reports can arbitrarily interleave with the periodic reports. For example, the sequence:

```
...
```

```
Sun Nov 5 13:10:55 MST 2017
o oldruns
Sun Nov 5 13:11:00 MST 2017
Sun Nov 5 13:11:05 MST 2017
Sun Nov 5 13:11:07 MST 2017
* driver.c
Sun Nov 5 13:11:10 MST 2017
* newqueue.c
Sun Nov 5 13:11:15 MST 2017
...
```

could be the result of a SIGUSR1 received at time :11:07 which resulted in reporting that file `driver.c` was modified between :11:05 and :11:07. The subsequent report about `newqueue.c` suggests that the file was modified between :11:07 and :11:10. *Note:* the modifications are always described with respect to the most recent previous report generated regardless of whether it was generated because of the periodic or the forced (SIGUSR1) reporting.

Technicalities

- When the program starts, it has no memory of any previous reports, hence it reports all the objects it finds in the directory as "created".
- If an object with name "name1" is removed from the monitored directory and a new one (of the same or different type) is created with the same name, it should, exceptionally, be reported as two separate and consecutive lines: one indicating removal (`-`) and the immediately next line indicating creation (of the appropriate type). Both those lines should include the name "name1" as usual.
- Receiving a SIGUSR1 should be handled and treated the same way as the triggering of the periodic report but it *must not reset* the timer for the periodic reporting. (See also the example above whether the periodic reporting occurred at :11:05 and :11:10, while the forced happened in-between at :11:07.
- If the directory indicated in *path* does not exist, the program exits immediately. Similarly, the program exits if at any point during execution the monitored directory is removed. The program must also exit gracefully (i.e., finishing and report printing underway) if it receives the `ctl-C` character (SIGINT signal).

Deliverables

You should submit your assignment as a single compressed archive file (`zip` or `tar.gz`) containing:

The `Makefile` needed to compile your program. The use of `make` and `Makefile` specifications will also be a requirement in subsequent assignments. Your `Makefile` should at the very least provide targets `diffdir379` and `clean`.

All of the source files needed to produce the `diffdir379` executable. Remember that this must be C code *only*, using the C99 style.

A short file `readme.md` (plain text or markdown markup) included with the archive to describe how you balanced the work across the two group members. Also indicate whether your code was tested (and running) on the VM OR OR on the physical hosts in lab CSC 219.

Monday, January 8, 2018