Course ‘Operating Systems Architecture’ – Project

UFAZ, L2
Lecturer: Pierre Parrend, Rabih Amhaz
Exercises: Pierre David, some from Vincent Loechner, Unistra.

1. Specifications

The objective of this project is to create a command `detect` to periodically launch a program and to detect state modifications. Your command shall admit following arguments:

```
detect [-t format] [-i interval] [-l limit] [-c] prog arg ... arg
```

Options are following ones:

- option `-i` gives the interval (in milliseconds) between to program launch (default value : 10 000 milliseconds) ;
- option `-l` gives the limit of launch number or 0 for no limit (default value : 0, i.e. no limit)
- option `-c` also detects modifications of return code, i.e. the argument of exit for the given program (default value : do NOT take return code into account) ;
- option `-t` provokes printing date and time for each launch, using specified format, compatible with library function `strftime` (default value : no printing).

The command `detect` launches the program specified by `prog` with following arguments.
At the first call, the standard output is printed (as well as its return code if option `-c` is activated).
After each call, the standard output (as well as its return code if relevant) is analysed. In case of discrepancy with previous output, the new standard output is printed.

2. Examples

In following example, the program `ls` is called with arguments `-l` and `/tmp` 4 times, with an interval of 10 seconds between two calls and the printing of time at each call. Whereas the standard output of the command changes, the new content is printed.

```
turing> ./detect -c -i 10000 -l 4 -t "%H:%M:%S" ls -l /tmp
14:47:21
total 3
--w------- 1 pda GG_MAI 5164 Jan 01 14:45 krb5cc_41866
--w------- 1 montavont GG_MAI 5204 Jan 01 14:45 krb5cc_76543
drwx------ 2 pda GG_MAI 4096 Jan 01 15:10 ssh-BkYk86pgQhHy
exit 0
14:47:31
14:47:41
total 2
--w------- 1 pda GG_MAI 5164 Jan 01 14:45 krb5cc_41866
drwx------ 2 pda GG_MAI 4096 Jan 01 15:10 ssh-BkYk86pgQhHy
14:47:51
```

```
turing>
```

In this example, one observes that on the first execution of ls, the initial result is printed: 3 files are found and the return code is null.
Then nothing happens during at last 10 seconds (the second iteration does NOT detect any modification). During the third iteration, at 14h47 and 41 seconds, a modification is detected since a file was deleted. The return call is not altered, so it is NOT printed again. During the fourth and last iteration, no modification is found.

3. Specific constraints and specifications

To simplify the implementation, in case of error, the execution will stop with suitable message.
Similarly, we ignore the standard error output of the launched program. Program stop for a reason other than `exit` is considered as an error.
To perform the waiting between two iterations, use library function `usleep`. Use function `getopt` to analyse options. Beware: implementation of `getopt` on Linux does NOT respect POSIX norme, we will thus add a « + » at the beginning of the argument chain (third argument of getopt) as indicated in Linux manual.
Please respect duly the printing model used in the example, so that tests can be automatized (see code on Moodle). Use NO intermediary file.
Of course, use only system primitives, unless for memory allocation, printing, option analysis, waiting and time formatting. For this project, we assimilate `execvp` to a system primitive.

4. Expected work

You will deliver

- Detect command
- Complementary test cases and a measure of code coverage, to ensure that your program is conform to specification
- A report describing the requested elements

You implementation MUST run on provided Debian virtual machine. You will strive to comply with the specifications and to prove it through your test cases, as well as to prevent any memory leak.
A archive is provided on Moodle with a `Makefile` and test cases.

5. Project output

The project is developed by groups of two students. Its deliverable must entail:

- Source files
- Project report as PDF formation, with in particular expected measures
- The set of test cases and the coverage measure obtained through `gcov`.

You will store your project, free from any binary file as a tar.gz archive on Moodle.