French-Azerbaijani University

Operating Systems Architecture

report on project

`Detect` command

Group:

CS-016

Authors:

Emin Sultanov

Fakhri Gulmammadov

March 21, 2019

Contents

1 Introduction **2**

2 Description **2**

3 Creation Process **3**
3.1 Steps of the Creation 3
3.1.1 Algorithm 3
3.1.2 C Code 3

4 `MakeFile`, test cases and result **6**

5 References **7**

1 Introduction

This report looks into the procedure of pipes, it helps understand clearly the usage of them, and the creation the options in c language as well as the working with the memory (allocation, comparing), time and reading/writing the files using POSIX library

2 Description

The main idea of the project is the working and creating the command that helps to periodically launch a program and to detect state modifications. This command will take as arguments the following options :
`detect [-t format] [-i interval] [-l limit] [-c] prog arg ... arg`
Where :

- option `-i` gives the interval (in milliseconds) between the every launch. By default the value of the interval will be 1000 milliseconds.
- option -l gives the number of initialisation of the program. By default this number will be equal to 0.
- option -c detects modifications of return code, it gives the exit value by the end of the program. By default value will be NULL (0).
- option -t makes the program display the date and time for each launch, using the format given by user. By default time format will be equal to NULL, and the code will not print the the date after launch.

After taking the options, `detect` will launch the program and the return code, if there will -c option. Afterwards, if there will launch limit, it look for the modifications of the output of the program and its return code, in case of any changes it modifies the previous standard output to the new one as well as the return code.

3 Creation Process

3.1 Steps of the Creation

3.1.1 Algorithm

There are five steps that have to do to make `detect` work properly:

1. The first step involves obtaining options and the program provided by the user. In this step, `detect` also should look for the errors if there is any of them.
2. After getting the options and the program itself, `detect` will launch the loop of the program execution, while decreasing the limit by one after every launch until the limit will not be equal to zero. In case if user did not give the limit it will be equal to zero, the program will be launched once.
3. When the program launched at least one time, its output should be saved in the memory to compare it with the current standard output. As it is not clear if the buffer, where our current standard output will be saved, will have enough place in the memory to keep it, it should be taken into account that there must be a possibility to increase the size of the buffer.
4. In case if user indicates `-c` option, `detect` should print the value on exit. It should print the return code if `detect` launched for the first time and if the code on exit is modified.
5. The last step to do is to display the time of the each launch of `detect`.

Additional step:

- To make display errors more organized, it is better to make function of all errors.

3.1.2 C Code

The second step of solving the problem is the creation of the program.

Libraries
It is necessary to include important libraries:

- `unistd.h` - for working with read/write functions of the POSIX.
- `stdlib.h` - for allocating a place in the memory.
- `stdio.h` - for creating errors and to flush the output .

- string.h - as our buffer will be a string and we need to compare the outputs.
- time.h - to display the time of the launch and for the checking if the time format is not exceed the size of time buffer.
- sys/wait.h - for displaying the return code on exit.

Errors

This function is useful in case of errors. There are 8 types of errors:

1. Error 1 - In case if the options are not appropriate or there is no option, i.e. if the user wrote -x option when there is no such option, it will print the right usage of the program
2. Error 2 - If the time format is too long it will print it.
3. Error 3 - In case if the interval is less or equal to 0, detect will display that the interval is not appropriate.
4. Error 4 - In case if the limit is negative it will display it.
5. Error 5 - In case if there is no argument after option, i.e. ./detect -c - i 1 -l 10, detect will print that there should be some arguments after.
6. Error 6 - if the pipe is not created
7. Error 7 - if the fork is nor created
8. Error 9 - if the buffer has not been reallocated

Init_time

This function is needed to display the time if there is any time format. To display time to the user it is necessary to create the time_ buf, to keep time value in it. To get the current local time and to write it into the buffer we need to use two functions :

1. time (NULL) - to get the current time
2. localtime - to make current time as local
3. strftime - to write the time in given format to the time_buf

Main

To implement the detect , firstly , there should be necessary variables :

- int p[2] - for making pipe
- char* format - to keep time format

- `int interval` - to keep the value of the interval between launch
- `int limit` - to keep limit of launchs
- `char returnCode` - to show the return code if there is option for it
- `int option` - for getting options from the function `getopt`
- `char* output` - standard output buffer
- `int prevCode` - for checking of modification of the return code

After defining the variable `detect` starts to get all the options given by user, while using `atoi` function to convert values of some options to integers or doubles.

```
while ((option = getopt(argc, argv, "+t:i:l:c")) != -1)
{
    switch (option)
    {
        case 'c':
            returnCode = 1;
            break;
        case 'i':
            interval = atoi(optarg);
            interval *= 1000;
            if (interval <= 0) error (3);
            break;
        case 'l':
            limit = atoi(optarg);
            if (limit < 0) error (4);
            break;
        case 't':
            format = optarg;
            break;
        default :
            error (1);
    }
}
```

Figure 1: Usage of getopt

When the options are obtained it is needed to create a pipe and fork for using `execvp` and send the result to the Parent process, where the current standard input will be read, while counting the read bytes, and saved in the `courant`, which is temporary buffer for standard output. In case if the size of buffer is less than needed `detect` increases it:

```
if (deja_lit >= taille - 1) {
    taille*=2;
    courant = (char*) realloc (courant, taille);
    if (!courant) error(9);
}
```

Figure 2: Increasing the buffer

Afterwards, it is important to compare the actual standard output with the current one with the help of `memcmp`, as it is possible to have EOF in the buffers. In case if the out-

puts are different detect rewrites actual standard output.

At the end, detect checks if there is any necessity to print return, in case if yes it also checks if the program launch for the first time (prevCode is equal to 210121) or the previous code is not the current one. Detect also displays the time if there is proper time format.

4 MakeFile, test cases and result

From the very beginning we were provided by several tests and MakeFile. These test were give to verify if our program works properly. With the help of MakeFile we can launch all the test and create the gcov to see which lines and how many times they were used. To create gcov we should write two commands in the terminal:

1. cc -g -coverage detect.c -o detect . After we should launch our tests.
2. gcov detecter.c

Example of usage of detect:
```
./detect -c i 1 -l 10 ls -l
```
Output :
```
total 660
-rwxrwxr-x 1 emin emin 17632 Mar 21 16:48 detect
-rw-rw-r-- 1 emin emin 5401 Mar 21 16:19 detect.c
-rw-rw-r-- 1 emin emin 1547 Mar 20 21:42 Makefile
-rw-rw-r-- 1 emin emin 896 Mar 20 23:07 test-100.sh
-rw-rw-r-- 1 emin emin 1048 Mar 20 23:51 test-120.sh
-rw-rw-r-- 1 emin emin 502 Mar 20 22:57 test-140.sh
-rw-rw-r-- 1 emin emin 845 Mar 20 13:39 test-160.sh
-rw-rw-r-- 1 emin emin 1395 Mar 20 13:39 test-180.sh
-rw-rw-r-- 1 emin emin 1126 Mar 21 16:18 test-200.sh
exit 0
```

5 References

List of web-sites that contains useful information to do this project:

- https://www.tutorialspoint.com
- https://www.stackoverflow.com
- https://linux.die.net/man/
- http://www.c-cpp.ru