Fundamentos de Sistemas de Operação

MIEI 2020/2021

Homework Assignment 1

Deadline and Delivery

This assignment must be performed ***individually*** by each student. Any detected frauds will cause failing the discipline immediately.

The code must be submitted for evaluation via the Mooshak using each student's individual account (http://mooshak.di.fct.unl.pt/~mooshak/). The solution must conform to the following rules: a) it must use multiple, concurrent processes; b) processes must communicate through pipes.

The deadline is ***23h59, October 23rd, 2020***.

Description

With this homework we want to obtain a system that will take a text, split it into individual words and, for each work, print the word's hash value followed by the word, as in the example below: [Note: bold represents user-typed input]

```
$ ./pipe_fork_exec
This is a text
9199165188857659392   This
9160321642071588864   is
6989586621679009792   a
9204794688391872512   text
Split into two lines!
9060450801897439232   Split
8051450971319959552   into
8065946932620558336   two
3456468733541744640   lines!
```

You may also run the program with an argument. If you have a text file named "`mytextfile.txt`" with the same text (the one printed in bold) as above, program output would be

```
$ ./pipe_fork_exec mytextfile.txt
9199165188857659392   This
9160321642071588864   is
6989586621679009792   a
9204794688391872512   text
9060450801897439232   Split
8051450971319959552   into
8065946932620558336   two
3456468733541744640   lines!
```

The system you are expected to build has three programs:

- `pipe_fork_exec` is the master program, setting the execution environment and launching the other two programs
- `split_words` is the producer: it reads the text file (or the standard input), splitting the lines into words and printing these words, one per line, to the standard output.
- `hash_words` is the consumer: it reads a line from its input (must be a single word), calculates an hash value for the word using the `unsigned long hashf(const char *str)` function, and then prints (in the standard output) both the hash value and the word, using the following instruction: `printf ("%lu\t%s\n", h, word);`.

As referred above, the `pipe_fork_exec` that sets the execution environment for collaboration and then launches the first two (`split_words` and `hash_words`). This `pipe_fork_exec` program must:

1. Create a UNIX pipe
2. Fork itself
3. The parent process will exec(ute) either `split_words` or `hash_words`. It will setup its output (or input) to (or from) the pipe and execute the appropriate program with `execv()`.
4. The child will handle the other program (either `split_words` or `hash_words`, depending on your choice in step 3 above). It will setup its input (or output) from (or to) the pipe and execute the appropriate program with `execv()`.

Figure 1 below illustrates this behaviour.

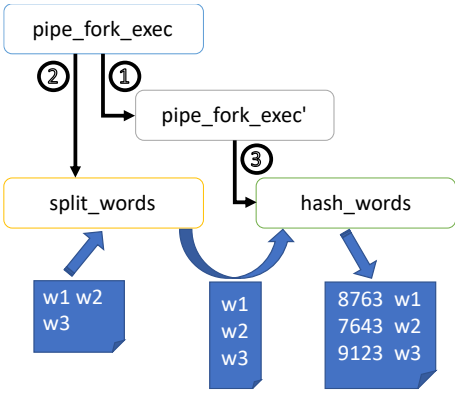

Figure 1: The software architecture and workflow

Work to do

We provide three C source files and one Makefile.

```
Makefile
hash_words.c
pipe_fork_exec.c
split_words.c
```

To compile, just enter the command "`make`". The programs will compile without errors but will not work. In each file there is a part of the implementation that is missing.

To complete each program, you will have to provide the missing code in the three files:

- `pipe_fork_exec.c` is missing the major part of the implementation of the `main()` function.
- `split_words.c` is missing a part of the implementation of the `split_print_words()` function.
- `hash_words.c` is missing the implementation of the `hash_and_print_words()` function.

Compile and Run the Application

To compile simply type

- `make`

To run you may (or may not) provide a command line argument:

- `pipe_fork_exec [textfile.txt]`

"Debug" the individual programs

If you follow (well, that is mandatory!) the architecture depicted in Figure 1, you can "debug" each program in isolation. For example, to debug `split_words`, you type

- `split_words`

and then type some text lines, each one terminated with a `NewLine` character; when you do not want to input more text, on a blank line, type the `Ctrl-D` character – the program should terminate cleanly. Or, you may prepare a text file, e.g., `split_test.txt` with some lines of text, and type

- `split_words < split_words.txt`

To debug `hash_words`, you use exactly the same strategy, but now you must provide a single word per line as input!

Submission

This homework assignment is individual and the code must be submitted for evaluation via the Mooshak system using each student's individual account (http://mooshak.di.fct.unl.pt/~mooshak/). You must submit the 3 sources files (`hash_words.c`, `pipe_fork_exec.c` and `split_words.c`).

IMPORTANT:

- **Do not wait for the last day to try to log into Mooshak;** the instructors may not be available to help you with your user/password problems!
- **There will be a limit of 10 submissions (per student) to Mooshak.** Do not use the Mooshak as a compiler, and remember that programs with warnings are NOT accepted, they are handled as erroneous programs.
- **If you do less than 10 submissions, your grade will refer to the last submitted version.**
- **If you attempt more than 10 submissions, your grade will refer to the 10th submitted version.**

The deadline for the delivery is ***23h59, October 23rd, 2020***.