EXERCISES — My redir

version #

ASSISTANTS C/UNIX 2022 <assistants@tickets.assistants.epita.fr>

Copyright

Contents

*https://intra.assistants.epita.fr

1 My redir

Files to submit:

- my-redir/my_redir.c

Authorized functions: You are only allowed to use the following functions:

- open(3)

Authorized headers: You are only allowed to use the functions defined in the following headers:

- sys/wait.h
- assert.h
- sys/types.h
- stddef.h
- unistd.h
- stdio.h
- errno.h
- sys/stat.h
- err.h

1.1 Goal

The goal of this exercise is to make you implement a program which mimics a Shell output redirection.

Be careful!

A guide is provided alongside the subject. You need to understand the following sections in order to complete this exercise:

- File descriptors
- open
- dup
- dup2
- fork
- exec

1.2 Definition

In Shell, an output redirection (`command > file`) redirects the standard output of a program to a file, the name of which is given on the right-hand side of the operator. If the file does not exist, it is created.

For instance:

```
42sh$ echo foo > foo.txt
42sh$ cat foo.txt
foo
```

1.3 What to implement

Now, to the heart of the matter. Your program will receive the following arguments:

- The first argument is a file name. If no file exists with this name, a new file must be created with permissions `0644`.
- The rest of the arguments make up the command to execute.

If too few arguments are provided, your program must exit with status `2`.

Your program must:

- execute the command and redirect its output to the file
- print on the standard output a message with a specific format

The message format is as follow (it ends with a newline):

```
<COMMAND_NAME> exited with <EXIT_STATUS>!
```

If an error occurs in the child process during the call to `execvp`, the child process must exit with status `127`. In this case, the parent process must exit with status `1`, and not print the exit status message. Otherwise, the command exits with `0` on success.

If any error occurs, you may print an error message on the standard error. Its content will not be tested.

For instance:

```
42sh$ ./my_redir foo.txt echo foo bar test
echo exited with 0!
42sh echo $?
0
42sh$ cat -e foo.txt
foo bar test$
42sh$ ./my_redir test.txt patatras # Non-existent command
42sh$ echo $?
1
42sh$ cat -e test.txt # File was created but is empty!
42sh$ ./my_redir ls.txt ls nonexistent # Non-existent directory
ls: cannot access 'nonexistent': No such file or directory
ls exited with 2!
42sh$ cat -e ls.txt
42sh$ # File exists but ls did not write anything to stdout.
```

Be careful!

If the given command is successfully launched but fails, this is not your problem: you must return 0.

It is my job to make sure you do yours.