Exercises — My pipe

version #

Assistants C/Unix 2022 <assistants@tickets.assistants.epita.fr>

Copyright

This document is for internal use at EPITA (website) only.

Contents

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

1 My pipe

Files to submit:

- mypipe/mypipe.c

Provided files:

- mypipe/mypipe.h

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

- pipe(2)
- dup2(2)
- close(2)
- waitpid(2)
- execvp(2)
- exit(3)

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

- err.h
- sys/wait.h
- sys/types.h
- unistd.h
- stddef.h
- errno.h
- string.h
- stdlib.h
- assert.h

1.1 Preamble

Be careful!

To complete this exercise smoothly, please do the my-redir exercise first.

Be careful!

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

- File descriptors
- File descriptors and fork
- dup

- dup2
- fork
- exec
- pipe

1.2 Goal

The goal of this exercise is to implement the function `exec_pipe` which pipes the output of a command into the input of another command, just like a shell pipe.

1.3 Definition

In `shell`, the `|` symbol ("pipe") is used to bind the standard output of the process on the left-side of the pipe to the standard input of the process on the right-side of the pipe for instance:

```
42sh$ echo Hallo | tr a e
Hello
```

1.4 What to implement

You will have to implement the function below:

```
int exec_pipe(char **argv_left, char **argv_right)
```

This function takes the arguments for each side of the pipe and returns the exit status of the process on the right of the pipe.

For example:

```
42sh$ echo Hallo | tr a e
Hello
```

Is the same as:

```c
// main.c

int main(void)
{
    const char *argv_left[3] = {"echo", "Hallo", NULL};
    const char *argv_right[4] = {"tr", "a", "e", NULL};
    return exec_pipe(argv_left, argv_right);
}
```

```
42sh$ make main
42sh$ ./main
Hello
```

It is my job to make sure you do yours.