Operating Systems Design Lab
Computer Engineering Department
Spring 2023/2024
Lab 3: Ordinary Pipes

Objectives

1. To understand the basics of ordinary pipes.

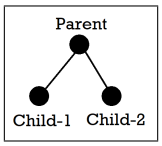
Prelab

1. Read chapter 3 of the textbook.
2. Read the manual pages of the following systems calls and functions.

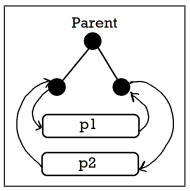
```
int pipe(int pipefd[2]);
ssize_t read(int fd, void *buf, size_t count);
ssize_t write(int fd, const void *buf, size_t count);
int close(int fd);
int strcmp (const char* str1, const char* str2);
size_t strlen(const char *str);
int toupper(int c);
```

Experiment

1. Write a simple program that implements the process tree below.



2. Update the program to make the parent process create two ordinary pipes before forking the children. Name the arrays of the two pipes `p1` and `p2`.
3. Update the program by closing proper file descriptors to ensure the following:
   (a) The parent does not read from/write to any of the pipes.
   (b) Child-1 writes to `p1` and reads from `p2`.
   (c) Child-2 writes to `p2` and reads from `p1`.



4. Update the program to ensure that each of the three processes work as described below.
   (a) **Parent**: it waits for the two children to terminate, and then terminates itself.
   (b) **Child-1**: it reads a word (a word has a maximum length of 10 characters) from the standard input. It writes the word to pipe `p1`. If the word is "exit", it terminates itself. Otherwise, it waits for a message from Child-2 via pipe `p2`. When the message is received, it prints it to the standard output and repeats the same procedure again.

(c) **Child-2**: listens to pipe `p1`. If the "exit" word is received, the process terminates. If different word is received, it converts all of its letters to capital form and returns the converted word to Child-1 via pipe `p2`. This process repeats indefinitely.