CS2518 - P2 - INTER PROCESS COMMUNICATION (IPC)

Overview

In this practical we are creating two independent processes that should be linked via a named *pipe*. A named pipe (also known as a FIFO for its behavior) is an extension to the traditional pipe concept on Unix and is one of the methods of Inter-Process Communication (IPC). The first process will be a c program used to read a file and the file content will be transmitted via the pipe to second program, a Python program, which will print out the content.

The Scenario

The first program called *writer* will be used to read the file *sensor_data.csv* from our first practical line by line. Then it will send each line via a named pipe to the second program. The second program will be a Python program called *reader.py* which will read line by line the data arriving via the pipe and print each line on the screen. Thus, we can create two programs (as two processes) using different programming languages that can work together.

Task 1: writer

The first task is to create the c program *write*. Create a file *write.c* which as a starting point could look like this:

```c
#include <stdio.h>
#include <stdlib.h>
#include <fcntl.h>
#include <sys/stat.h>
#include <unistd.h>
#include <string.h>


int main() {
    printf("I am an empty c program!\n");
    // open file
    // create the pipe
    // read file line by line and write to pipe
    // close the file, close the pipe
    return EXIT_SUCCESS;
}
```

You can compile this program by using the following command:

```
gcc writer.c -o writer
```

Now add the following functionality to the program:

(1) Open the file so that the program can read it. The function *fopen()* can be used for this purpose.
(2) Then create the pipe using *mkfifo()*. This creates a pipe which you can find within your file system (have a look).
(3) The pipe then needs to be opened for writing using *open()*. Note that this call will be blocking until the other program connects to the pipe for reading. So you might have to comment out this step to develop the next pieces (or develop Task 2 in paralell).
(4) Now read the file line by line. This can be done by using *fgets()* in a while loop.
(5) Within your loop for every line use *write()* to send the data via the pipe.

Task 2: reader

We now need the program *reader.py* for the other end of the pipe; a python program that connects to the pipe and reads line by line what *writer* is sending. You would need in your program *import os* so that the functionality to access the pipe is available in Python.

Add the following functionality to the program:

(1) Check if the pipe exists using *os.path.exists()*. We can do this as the pipe is a construct represented via the file system. If the pipe is not present, you could create it using *os.mkfifo()*
(2) Now open the pipe for reading using *open()*
(3) Read and print each line ...

Task 3: Optional

Replace the program in Task 1 by an equivalent Python program and replace the Python program in Task 2 by a c program.

CS2518 Continuous Assessment - PART 2

Please submit the two commented programs you created (Task 1 and Task 2). In addition, answer the following question: *Instead of writing and reading to a pipe we could write and read to a file. What would be the difference? Why would be a pipe a better choice compared to a file?*

The programs and your answer has to be submitted at the end of the semester together with questions from the other practicals (You do not have to complete all tasks/answers within the practical slot).