CSE 5031 Operating Systems
2020/21 Fall Term

Project: 3
Topic: Process Management and IPC
Date: 14.11 - 21.11.2020

Objectives:

- To develop a multi-process application that shares a program.
- To implement inter-process communication using ordinary pipes

References:

- **The GNU C Library Reference Manual** (http://www.gnu.org/software/libc/manual/pdf/libc.pdf)
- **Linux The Textbook. S.M. Sarwar, R.M/ Koretsky. CRC Press 2019. ISBN 978-1-138-71008-5**
- **Linux System Programming 2d ed., Robert Love, O'Reilly 2013**

Section A. Project Definition

A.1 Multiprocess Application and IPC Framework Definition Notation

Following table summarizes the schematics and we use to depict the organization of multiprocess applications and related inter-process communication framework.

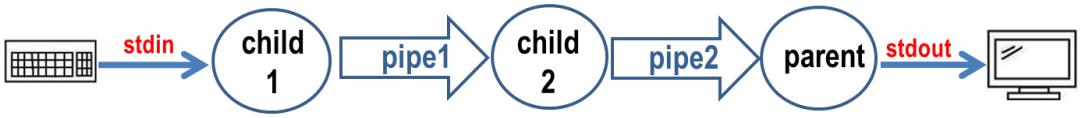
✓ **Processes** are shown with labelled circles, indicating their hierarchy in the process tree e.g. Parent, Child-1, .. Child-n; the name of the program they run e.g. "bash", "a.out" etc.	
✓ **Standard input**, **output**, **error** file descriptors (low level API handles) are shown with a labelled **narrow arrows** pointing in data flow direction.	
✓ Example of a process reading from standard input associated with keyboard, and writing to the **standard output** associated with a terminal or terminal window**.**	
✓ Ordinary **pipe**, the unidirectional communication channel, is depicted with a **large arrow**. Arrow head pointing in data flow direction. A pipe may be labeled with the **file descriptor array** name e.g. **pfd [2]** (**p1** writes to the pipe using pfd[1] whereas **p2** reads from the pdf[0])	
✓ Redirection of **standard input-output-error** file descriptors to a **pipe's** end is drawn with connected large and narrow arrows. The schema depicts the redirection of the child's **stdout** file descriptor to the write-end of the pipe, the parent reads from pipe's read-end	

A.2 Project Coverage

In this project you will develop an application consisting of **3 processes** that share the **same program** and communicate through **2 ordinary pipes**.

The application and inter-process framework depicted here after describes the scenario in which:

- → the process **child-1** reads data lines from **standard input** associated with the keyboard; process and writes them to the **pipe-1;**
- → the process **child-2** reads from **pipe-1** data generated **by chid-1** and writes them to the **pipe-2;**
- → the **parent** process reads from **pipe-2** data generated **by chid-2**; and writes the result to **standard output** associated with a terminal window**.**

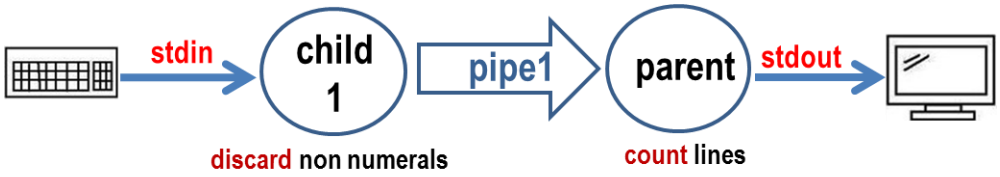


Section B. Implementing Multi-process Application

You will implement the multi-process application defined earlier in **two phases** to ease its implementation and debugging procedures.

B.1 Phase I. Deploying a 2 Processes Framework

As the starting step implement just **2** processes: the **parent** and its **child** sharing the same program and communicating via an ordinary pipe as depicted here after.



i) **Child1 process**:

- ✓ **reads** a varying length input string from **keyboard** until the **end of file** character is entered (left ctrl+d);
- ✓ **discards** records that contain non numerals (ASCII code not in the ['0' .. '9'] interval);
- ✓ **writes** verified string to "**pipe1**", including the new line '**\n**' character that marks the end of the record;
- ✓ **closes** write file descriptor of "**pipe1**" when **stdin** **end of file** is detected signaling the end of data input.

ii) **Parent process:**

- ✓ **creates** the IPC channel "**pipe1**" and the child process;
- ✓ **reads** a varying length string from "**pipe1"** in a buffer of max. 40 bytes, until it is closed (**end of file);**
- ✓ **counts** the number of input lines;
- ✓ when "**pipe1**" is closed, displays the line count on **stdout**;
- ✓ **waits** for the **child** to end;
- ✓ **closes** read file descriptor of "**pipe1**" then terminates.

Implementation Specifications:

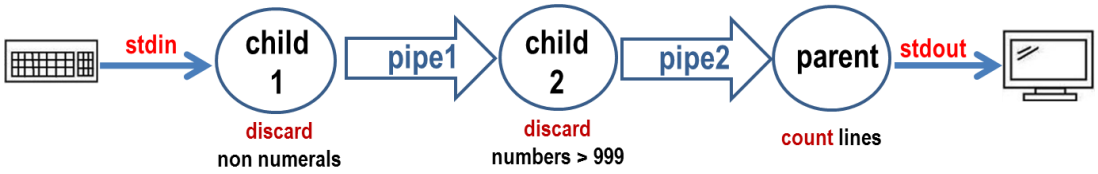
- ✓ Both processes are expected to **close** unused end of the **pipe** before starting processing.
- ➢ Use "**fgets**" **I/O stream** function to read varying length records buffer of maximum 40 bytes from **stdin** (refer to the GNU C Library Reference Manual for the parameters and returned results).
- ➢ To avoid trivial indexing mistakes use constant definitions such as **RD** and **WR** to express the file descriptor for a given **pipe**

```
#define RD 0
#define WR 1
……..
```

- ✓ **read** (**pipe [RD]** , buf, len); …or **write** (**pipe [WR]** buf, len);
- ✓ **close** (**pipe [RD]**);

B.2 Phase II. Deploying 3 Processes Framework

Once phase I is operational expand the application to work with 3 processes implementing the **IPC** framework and functionality depicted here after.



i) **Child1 process** performs the procedure defined in section B.1 as is.

ii) **Child2 process**

- ✓ **reads** varying length string from "**pipe1"** in a buffer of max. 40 bytes, until the **end of file;**
- ✓ **if** the integer string consists of less than 4 digits (not in the range [0 ..999]) **writes** the input string to "**pipe2**", including the new line '**n**' character;
- ✓ when "**pipe1**" is closed
  - o **closes** the read file descriptor of "**pipe1**";
  - o **closes** the write file descriptor of "**pipe2**" signaling the end of data input to the parent.

iii) **Parent process**

- ✓ **creates** the IPC channels "**pipe1**" and "**pipe2**"; and child processes;
- ✓ **reads** a varying length string from "**pipe2"** in a buffer of max. 40 bytes, until the **end of file;**
- ✓ **counts** the number of input lines;
- ✓ on "**pipe2**" **end of file**, displays the line count on **stdout**;
- ✓ **waits** for its **children** to end;
- ✓ **closes** its file descriptors then terminates.

Section C. Project Report

Do not submit a result if your program does not work as specified.

- ✓ Name your **source code** as **prj3.c** and add a comment line consisting of your name and student-id.
- ✓ Store either the **prj3.c** in the "**Prj3**" folder, located at the course web site under the tab **CSE5031 - OS Section - X/Assignment**; where "**X**" stands for (1,2,3,4) your laboratory session group.

Warning

You are encouraged to discuss the implementation procedures and general concepts behind the projects with your fellow students. However, **plagiarism is strictly forbidden**! Submitted report should be the result of **your personal work**!

Be advised that you are **accountable** of your submission not only for this project, but also for the mid-term, and final examinations. Your project grade may be reevaluated retrospectively, had you fail to answer correctly the same or a similar examination questions that you have solved with success in your submissions.