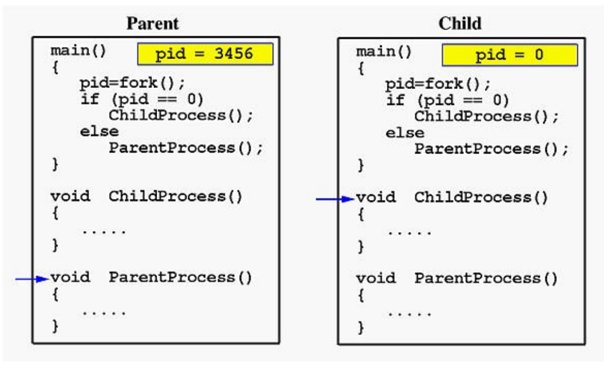
KU LEUVEN

GROUP T LEUVEN CAMPUS

# Operating Systems

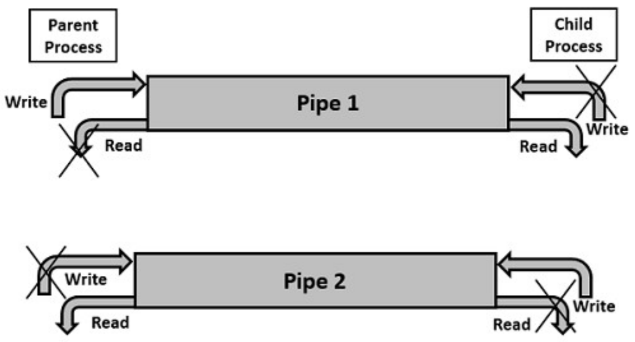
# Project labs: step 2

Milestone 2: A logging process with IPC



```
main()        pid = 3456
{
   pid=fork();
   if (pid == 0)
      ChildProcess();
   else
      ParentProcess();
}

void  ChildProcess()
{
   .....
}

void  ParentProcess()
{
   .....
}
```

```
main()        pid = 0
{
   pid=fork();
   if (pid == 0)
      ChildProcess();
   else
      ParentProcess();
}

void  ChildProcess()
{
   .....
}

void  ParentProcess()
{
   .....
}
```



Bachelor Electronics/ICT

Course coordinator: Bert Lagaisse
Lab coaches:

- Jeroen Van Aken
- Yuri Cauwerts
- Ludo Bruynseels

Last update: November 13, 2022

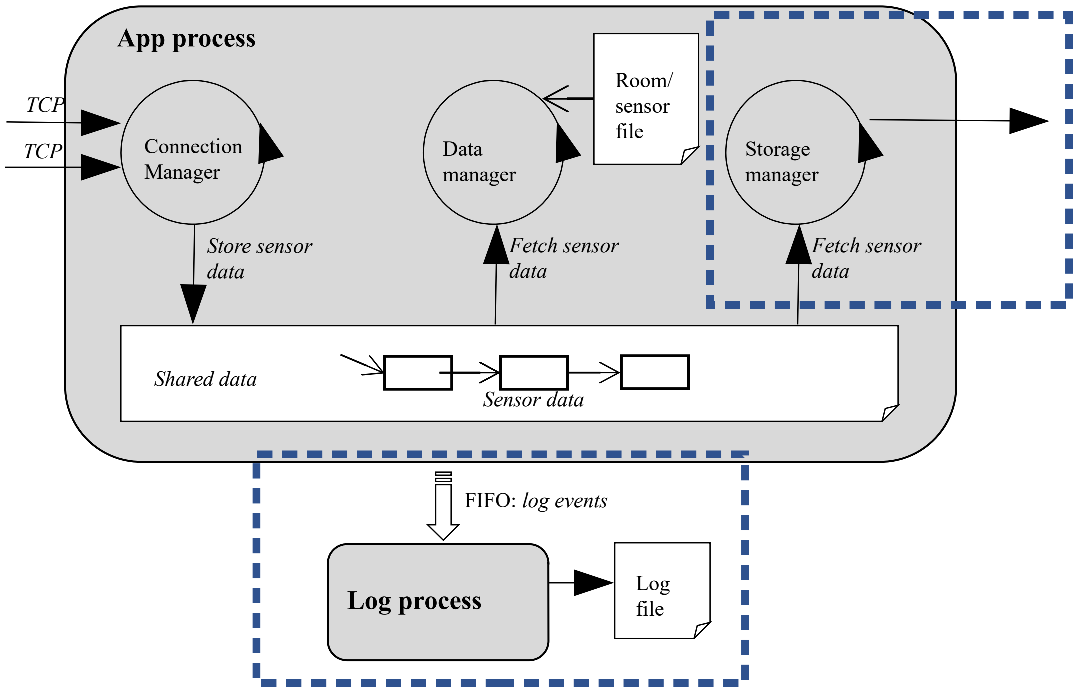
## Project labs: step 2 – Inter process communication

**Lab target 1: Learn how to spawn processes using fork and implement inter-process communication using pipes.**

**Lab target 2: Applying the concepts of forking and pipes to the logging process in the project. You will need to submit this as Milestone 2 on Toledo.**

### *Project overview*

*The relationship of this lab to the final assignment is indicated by the dashed blue line.*



### *Warm-up Exercise 1: appending sensor data to a text file.*

In the previous lab we learned how to perform file I/O in a C program. As a quick rehearsal, and as a quick warm-up for this session, we will first create the storage manager of the project. We provided a *config.h* with the data types and structures.

Create a test program (*main.c*) and sensor_db module (the storage manager) to store temperature data into a text file, structured as a comma separated value file (csv). Implement both the header (*sensor_db.h*) and the implementation (*sensor_db.c*) with the following operations:

- an operation to open a text file with a given name, and providing an indication if the file should be overwritten if the file already exists or if the data should be appended to the existing file.
- an operation to append a single sensor reading to the csv file
- an operation to close the csv file.

The header file sensor_db.h should at least provide the following operations:

```
#include <stdio.h>
#include <stdlib.h>
#include "config.h"
#include <stdbool.h>

FILE * open_db(char * filename, bool append);
int insert_sensor(FILE * f, sensor_id_t id, sensor_value_t value, sensor_ts_t ts);
int close_db(FILE * f);
```

## *Exercise 2: creating a child process and communicate with it*

Develop a C program using ordinary pipes in which the parent process sends a string message to a child process, and the child process reverses the case of each character in the message and then prints it on the command-line. For example, if the parent process sends the message "Hi There", the child process will print "hI tHERE". You can use the pipe() command for this.

**Hint 1:** Use the fork() and pipe() functions as illustrated in the lectures and book.
**Hint 2:** You will need some interesting functions like strlen(), islower() and toupper().
**Hint 3:** You can assume a maximum length on the buffer that contains the text message.

## *Milestone 2: creating the log process of the project*

**Reminder: Upload your solution for this milestone on Toledo as 1 zip!**

Use the zip build target in the make file to create milestone2.zip: **`make zip`**

With the knowledge of the 2 exercises above, we now create the logger of the project. Create a program such that a new log process is created as a child process, and a communication pipe between the storage manager and the newly created log process is installed. The log process receives log-events from the storage manager and writes them to a text file (the log file).
A log-event contains an ASCII info message describing the type of event. A few examples of log-events are:

- A new csv file is created or an existing file has been opened.
- Data insertion succeeded.
- An error occurred when writing to the csv file.
- The csv file has been closed.

For each log-event received, the log process writes an ASCII message of the format <sequence number> <timestamp> <log-event info message> to a new line on a log file called "*gateway.log*".
Try to provide a clean shutdown for both processes. Make sure you don't end up with orphan or zombie processes.
We will compile and test your solution against the header *sensor_db.h* specified above, check if the logger is running as a separate process, and check if your gateway.log contains our test operations.