# 6ELEN013W Operating Systems and Drivers

# Coursework 2 – Programming Assignment

Submission deadline: Thursday 9th April 2020 by 13:00 hrs via Blackboard

# Specification

As explained during the lectures and tutorials, you are to write a program in C, based on the tutorial exercises, that performs the following:

## Parent Process

* A file “STATUS.TXT” is created to contain information about the running of the program which should include:
  + The time the program was run
  + Process and Child process information (DON’T do this from the child process)
  + You are also to write various pieces of information associated with the inode of this output file.
  + Use printf to write to the file (this will work if STDOUT has been redirected)
* Maps the standard output to the file STATUS.txt
* The process creates one child
* Every time CTRL-C is entered, the signal SIGUSR1 is sent to the child
  + You will need to do this using a system call
* When the child terminates, to write to the STATUS file the time of the program termination
  + You will need to use a suitable signal to know when this happens

## Child Process

* It is to run for 5 seconds or until CTRL-Z is entered – manage this using a suitable signal
* Each time the signal SIGUSR1 is received, a random number between 0 to 9 is generated and then sends it to a python driver/connector.

## Python connector

* The python code is to be a connector/driver to the database
* Each time the code receives a number from the child, to add it to the database
* When the timer expires or CTRL-Z is pressed in the child process (whichever occurs first), the contents of the database are to be retrieved via a python driver/connector and written to a RESULTS.TXT file.
  + It should only print out the randomly generated numbers, not the unique auto number field

## Database Notes

* Should only contain two fields
  + Auto number – unique id for record
  + The randomly generated numbers from the C program
* You can pre-create the database separately, it is not necessary to create it from your C or Python code. You just need to ensure that it is emptied when the results are retrieved from the child process

## General Specification

* Use python calls within your C code - it requires python.h
* Use python 3.8.1
* Checks should be done on relevant function calls to ensure no errors and that the operation was successful. There are a number of functions that can return -1.
* Proper housekeeping should be done when processes terminate (e.g. closing/releasing resources)
* Ensure that printf calls are written to files and not to the console.
* Include your student ID as a comment in your code

## Notes

* There will be at least 4 signals in the program.
* A single signal handler should be written to manage the signals for each process
* Marks will be awarded for presentation (includes comments).
* Ensure that the signals do not clash between processes
  + The processes should only respond to the signals described in their specification
  + You can do this by ignoring the appropriate signals from within either the parent or child process.

# Coursework 2 Mark Allocation Scheme

|  |  |
| --- | --- |
| **Criteria** | **Marks** |
| Parent   * Status file contents * DUP2 to Status file * Fork * Manage CTRL-C * System call to Child with SIGUSR1 * Manage child complete * Child complete Action | 6  4  4  4  8  4  4 |
| Child   * Manage time limit using signal * Manage CTRL-Z * Manage SIGUSR1 * Generation of random numbers * Call python (C to python interface) | 4  4  4  4  6 |
| Python   * Database structure and operations * C to Python add random number to database * Process data from database at end * Write results to file | 4  4  6  6 |
| General   * Ensuring operation checks are performed to ensure that they succeeded (e.g. fork, dup2, signal, fopen, etc) * Good layout of code – appropriate indentation * Technical commenting of parts of code * Ignoring relevant signals | 6  6  6  6 |