Operating Systems

6CM503

Computer GE 645 with OS Multix and its creators around

02 Assessment Brief 2025/26

Linux System Calls: Client-Server Design using Inter-Process Communication (IPC)

Dmitry A. Zaitsev

1. Module Leader

- Dr. Dmitry A. Zaitsev
- Email: d.zaitsev@derby.ac.uk, Phone : 01332592678
- Office: MS229. Recommended days for individual communication: Monday, Wednesday.
- Website: https://www.derby.ac.uk/staff/dmitry-zaitsev

2. Key dates and details

Assessment Detail	Assessment Information
Assessment Type:	Individual Report, Program Code
Assessment Weighting:	30%
Word count/Length:	1,000 words / 3-4 pages (without listings)
Learning Outcomes:	1, 2
Submission Method:	Blackboard Assignment
Submission Date:	17:00 Noon Greece time, 19/01/2026
Provisional Feedback Release Date:	17:00 Noon Greece time, 09/02/2026

3. Description of the assessment

System calls represent one of the basic interfaces of OS, together with the user interface and the interface with computer hardware. Traditionally, system calls are implemented via a program interrupt to provide not only a call of the corresponding OS kernel routine but also the switching of the processor context and protection mode from User to System. We can use system calls directly via a single library routine and a block of parameters; also, standard libraries of programming languages provide a series of convenient functions that represent wrappers for the corresponding system calls.

The goal of the assessment is to master the system call interface of Linux and C programming using system calls, especially to implement an application as a set of communicating concurrent processes. We focus on composing programs in command-line utility-like style and using such basic inter-process communication facilities of Linux as pipes, semaphores, messages, and sockets, including TCP/IP sockets.

With this assessment students master skills of using the OS API of system calls, focusing on Inter-Process Communication via pipes, semaphores, and messages – locally, and sockets – for worldwide communication within a TCP/IP network. A student is assigned by 1 of 4 mentioned facilities and implement the required data processing as a separate process, a kind of server, with parameter transmission via given IPC by a client. Along with the report, students submit code of client-server applications for given IPC. There are 12 individual tasks to develop and debug software using OS IPC system calls.

The assessment involves an incremental learning approach facilitating a better student experience specified in the corresponding section of the Laboratory and Home Training where, at first, a utility program with a command line interface and options is developed, resembling the system utility interface; second, students run examples provided for each kind of IPC. Finally, a client-server

application is developed for a given IPC within the following list: pipe, semaphore, message, and sockets. The server program provides the same information processing as the utility program; input parameters and results are interchanged with the client program using given IPC. The report supposes analysis of using communication facilities supplied with graphs, recommendations, and conclusions.

Basic information:

- Lectures 6-12
- Laboratory and Home Training, section 2, weeks 7-12
- OS-individual-task-number-assignment

4. Relationship to Programme Assessment Strategy

The second task of the module Coursework four portfolio tasks.

5. Attributes and Skills

The attributes and skills are listed in the following table.

	Skills	Links to useful resources
☒	Critical thinking	Analysis of historical and contemporary OSs.
☒	Communication	Working within common Linux server environment.
☒	Collaboration	Partial teamwork on the same task variants.
☒	Creative problem solving	Script, utility, and kernel loadable module design
☒	Self-direction & planning	Planning efforts and activities to satisfy task submission deadlines.
☒	Numeracy, statistics & financial literacy	Top X lists and statistics on most widespread OSs for specific application domains, awareness of financial aspects with OSs' comparative cost and reliability
☒	Digital	Students not only use but create digital technologies

	Skills	Links to useful resources
☒	Resilience	Students learn how to avoid stress inflicted by with necessity to satisfy deadlines and recover after busy times
☒	Adaptability	Adaptation to using rather novel, for students, type of OS compared to officially supported by UoD MS Windows
☒	Leadership & future thinking	Developing kernel loadable modules, students think about OS of future

6. Assessment Content

Developing big application systems raises a problem of their reliability. Adding components to a single application ensures feature interaction problems. Because of it, most big systems are implemented as a set of communicating processes using OS IPC for information exchange and synchronization. Operating systems use IPC for system process communication as well. IPC also provides communication of processes over a network involving client-server interaction.

The central task of the present assessment is a client-server system design using given OS IPC according to the student's individual task.

6.1. Brief description of the assessment workflow

Design a client-server application using given OS Linux Inter-Process Communication (IPC) facilities for information exchange, processing, and synchronization between concurrent processes.

Recommended workflow:

1. Design a client and server to provide service according to your individual task using your assigned IPC from the list: pipes, messages, semaphores, sockets.
2. Run and test client and server utility programs, and collect evidence that they are running within the Linux environment and that the required IPC constructs have been created.
3. Explain in detail how client-server programs work, focusing on information exchange using graphical diagrams of communication.

Individual tasks: Section 6.2.

Content of report: Section 6.3.

Recommendations on IPC implementation: Laboratory and Home Training, section 3.

Example program code – module supplemental materials – examples of client-server pairs implementation using: pipes, messages, semaphores, and sockets.

6.2. Individual tasks

a. Function:

1) Concatenate two strings
2) Find a substring
3) Replace a substring
4) Compare two strings
5) Reverse a string
6) Swap two parts of string
7) Extract a substring using specified position and length
8) Remove letters specified in the second string
9) Insert a substring from the specified position
10) Remove a substring
11) Remove the inner 1/3 part of string
12) Remove the outer 1/3 parts of a string

b. IPC facility:

1) *Pipes*
2) *Messages*
3) *Semaphores*
4) *TCP/IP sockets*

- **Individual task number is pre-assigned according to the file of individual task numbers (OS-individual-task-number-assignment.pdf); please use column 2 for the number *a* and column 3 for the number *b*, choose the row based on the student name and surname, specified in column 1**

6.3. Content of the assessment report

1. How to run the client and server
2. Description of client and server illustrated with graphical schemata and snippets of code
3. Graphical scheme of communication for a given IPC inscribed with used functions
4. Results of the client-server application testing (no less than 3 tests) and using IPC

The report should be supplied with an archive containing (in textual files ready to run) client and server programs, supplied with comments on how to use them, and evidence of their testing.

7. Assessment Rubric

Criteria	Excellent (70-100%)	Very good (60-69%)	Good (50-59%)	Satisfactory (40-49%)	Unsatisfactory (<40%)
IPC application design, 60%	Successful implementation of all requirements	Successful implementation of basic requirements	Successful implementation of basic requirements with minor	Implementation of basic requirements with some errors	Basic requirements have not been implemented

			errors		
Tests and evidence of using IPC, 20%	Comprehensive successful tests and evidence	Successful tests and evidence for basic functions	Tests and evidence with some imperfections	Tests and evidence with some with some errors	No tests passed or no evidence provided
Report writing, 20%	The report is exceptionally well-structured, clearly written, and professionally presented.	The report is well-structured and clearly written with minor errors.	The report is complete but may lack clarity or detail in some sections.	Report is poorly organized or lacks clarity. Analysis is weak or incomplete.	Report is disorganized, difficult to follow, and lacks key elements.

8. Assessment AI-assistance

In this assessment AI-assistance is permitted in the following ways:

- Explanation of IPC facilities
- Check of obtained results

In this assessment AI-assistance should not be used for:

- Code composition
- Writing report

It is YOUR RESPONSIBILITY to check ALL information generated by generative AI tools. Any misuse of generative AI tools could be considered ethical academic misconduct (as per Academic Regulations, Academic Misconduct, Section J2). If in doubt, please consult your module leader.

9. Anonymous Marking

You must submit your work using your **student number** to identify yourself, not your name. You must not use your name in the text of the work at any point. When you submit your work in Turnitin you must submit your student number within the assignment document and in the *Submission title* field in Turnitin. Guidance is available showing how to do this.

10. Assessment Regulations

The University's regulations, policies and procedures for students define the framework within which teaching and assessment are conducted. Please make sure you are familiar with these regulations, policies and procedures.