Cairo University
Faculty of Engineering
Computer Engineering Department

OS Synchronization

Document Structure

- Objectives
- Introduction
- Requirement
- Guidelines
- Deliverables

Objectives

- Implement process synchronization in the context of a classical problem.
- Practice the use of IPC techniques.

Platform *Linux*

Language *C*

Introduction

Read the problem statement and how to solve it in the book "Operating Systems Design & Implementation" 3rd edition, section 2.2.4.
After reading it, you should know that your program should have a producer and a consumer. The producer produces items and places them into a bounded buffer as long as it is not full, and the consumer consumes the items as long as the buffer is not empty. You are required to implement this using inter-process communication.

Requirement

The producer and the consumer are 2 separate programs each of them has its own file and can be executed on its own, i.e., the producer and the consumer can be run separately in two terminals.

The Producer

- If the buffer is full, it waits (blocks) for the consumer to consume an item.
- Otherwise, it produces a new item and adds it to the buffer.

The Consumer

- If the buffer is empty, it waits (blocks) for the producer to produce an item.
- Otherwise, it consumes an item from the buffer.

Both

- Each should print a message before they try to produce/consume an item.
- Each should print a message after they have produced/consumed an item (this message should include the item).
- The *rate* of production/consumption (items per second) for each should be passed to each one independently as a command-line argument and it should NOT be shared between them.

Guidelines

- Read the document carefully at least once.
- The item can be anything distinguishable, e.g., the size of the buffer at time of production. You can use another form of items but after taking permission from your TA.
- The bounded buffer is a shared memory block.
- The user should be able to enter the size of the buffer.
- The shared memory must be protected with semaphores.
- Refer to the labs for details about message passing, shared memory and semaphores.
- Your program must not crash.
- You need to release all the IPC resources upon exit.
- You can use any IDE (Eclipse, Code::Blocks, NetBeans, KDevelop, CodeLite, etc.) you want of course, though it would be a good experience to use make files and standalone compilers and debuggers if you have time for that.
- The code should be clearly commented and the variables names should be indicative.

Deliverables

You should deliver two code files; *producer.c* and *consumer.c*.