Memory Management algorithms Evaluation

Savvas Rostantis

Contents

Project Overview

The project contains two folders: Generator and Manager. The first contains Generath.h, poisson.h, Register_table.h, Operation.h, Virtual_Process.cpp.h for header files and main.cpp files, Generath.cpp, poisson.cpp, Register_table.cpp, Operation .cpp, Virtual_Process.cpp. for executable, and the second Memory_Manager.h, Waiting_List.h, Operation.h, Memory_Algorithms.h for header files main.cpp, Memory_Manager.cpp, Waiting_List.cpp, Operation.cpp, Memory_Algorithms.cpp for executables, correspondence with the implementation of the first and second terminals. The Poisson.cpp file contains the implementation of the algorithm to generate exponentially distributed Poisson time messages.

Execution

The files in the Generator folder have a makefile, executing with the command:

- ./Memory_Placement [duration] [lo] [li] [t] [T] [size].

The files in the Manager folder have a makefile, running with the command:

- ./Memory_Placement [size] [duration].

Processes Overview

This project consists of four processes:

1. generator: The first terminal, generates processors.
2. manager: The second terminal, memory organization.
3. Organizer: The first system, waiting list control.
4. vp_processes: The first terminal, processes.

Processes communicate with each other through four shared memories where they synchronize with floats. One memory is for Manager Generator communication, the other for Manager Organizer Generator communication, one for terminating processes and the other for restarting those inactive processes.

Program Description

The generator generates exponentially distributed virtual processes and sends its data to the manager. The manager checks if it fits in the memory and if so saves it to a structure that has the id and memory endpoints. Then it updates the generator and writes it to the register structure. If not, the manager puts it on the waiting list. The generator reduces the idle and run time of the process by one second each time and performs again accordingly. Every second it checks if someone comes out of the waiting list and repeats. Depending on the generator's information, the manager puts - extracts processes from memory according to the given memory placement algorithm. Then data is written into three files for memory, end state and current states. The organizer is responsible for informing the generator from the manager as soon as the process is out of the waiting list, there is no reason to block communication.

Tested environment

The implementation language is in C ++ and was developed on Linux Ubuntu 15.10.

Execution example