Project 2 Memory Simulation

CSCE 4600 Operating Systems

Faris Hawamdeh

Dustin Fennessy

Eric Guzman

d

D

# Introduction

The purpose of this project was to develop and employ memory management systems that handle arriving processes. The processes were generated from our process generator which we used in Project 1. There are 50 processes for each run, with the arrival time every 50 cycles. These processes are assumed to run to completion, however, they can only run if they have enough memory.

# Memory Systems

Our memory management systems are written in two ways. Both of them are valid in that they work, but this project attempts to delve into the differences between the systems.

## Malloc()/Free() Management System

The first memory manager of our project

* Has a memory pool of 10MB
* Will only run a process if sufficient memory is available
* Uses calls malloc() and free() to allocate and de-allocate memory
* Relies on OS-specific systems calls, so performance may vary between OS’s

## Custom Management System

The second memory manager of our project

* Has a memory pool of 10MB
* Will only run a process if sufficient memory is available
* Uses custom-built MyMalloc and MyFree functions to allocate and de-allocate memory
* Does not rely on OS system calls
* Employs the buddy system in its implementation

# Test Environment

Our tests were run on the CSE machines. The general system specs for the machines are below.

* Run on Linux
* AMD 2200Mhz processor (with 1-3 cores)
* Barebones from user side
* g++ for compilation

# Experiment

In order to test the efficiency of the two memory management systems, we run our program 5 times for each system. We compare data such as total process time and cycle distributions between the two systems. The results of our experiment are posted below along with data charts and summaries to explain some of the aspects of the comparisons.

## Total Process Time Comparisons

In the table below we show the comparison between the process times between malloc/free vs MyMalloc/MyFree. As you can tell, malloc/free are fairly consistently more efficient.

# Appendix