Faculty of Engineering and Applied Science

SOFE 3950 - Operating Systems

Lab 4: HOST Dispatcher Shell

Massimo Albanese 100616057

Lyba Mughees 100750490

Daanyaal Tahir 100746644

Jean-Paul Saliba 100741759

March 26th, 2023

Introduction

The Hypothetical Operating System Testbed (HOST) is a multiprogramming system with a four level priority process dispatcher operating within the constraints of finite available resources. Our system is designed to have 1024 megabytes of ram, used along with 2 printers, 1 scanner, 1 modem, and 2 CD drives. The dispatcher has 4 job queues with decreasing priority: realtime, then first, second, and third user priority. While it always takes jobs with higher priority, lower priority jobs can be inserted between higher priority jobs to create a round-robin system.

Memory Allocation

Our dispatcher has 1024MB of memory, which is divided into 2 portions: 64MB is assigned realtime memory, while the remaining 960MB is shared among all user jobs. Our dispatcher uses the First Fit memory allocation scheme to allocate contiguous blocks of memory, as it is the fastest to search and easiest to implement. For our application, the overhead of First Fit at the beginning of each allocation is negligible.

Structure and Function

Queueing:
The queue is implemented using linked lists for the process control block (PCB) and the memory structure. The queue is FIFO (First In First Out), and has three functions:

- push: adds a new node to the end of the queue
- pop: pops the first element of the queue

Process Control Block (PCB):
The PCB contains all information about the process, such as arrival time, priority, CPU time, memory and resources required, and status. It is used by the dispatcher to assign the process to a queue accordingly. It has 3 functions:

- load_dispatch: parse the input file and load the process information
- load_jobs: load each process into the queue based on arrival time and priority
- queues_empty: checks if the queues are empty

Memory Management:
There are two functions used to manage memory:

- alloc_memory: allocate memory to realtime or shared memory pools based on process
- free_mem: free the allocated memory when the process exits

Resource Management:
Resource management is used to check and allocate resources based on process' needs and availability. There is one resource management function:

- resources_available: checks if the needed resources for a process are available for allocation