Operating Systems Design Lab
Computer Engineering Department
Spring 2023/2024
Lab 8: Memory Allocation

Objectives

1. Write a C program to simulate the following variable partition size memory allocation algorithms:
   (a) **Best-fit**: chooses the smallest memory hole that is big enough to accommodate the request.
   (b) **Worst-fit**: chooses the largest memory hole that is big enough to accommodate the request.
   (c) **First-fit**: chooses the first memory hole that is big enough to accommodate the request.

Prelab

1. Review Chapter 8 of the textbook.

Experiment

Write a program that takes the following as an input:

1. Number of processes N.
2. The size of the total memory in bytes.
3. The scheduling algorithm to use (1: Best-fit, 2: Worst-fit, 3: First-fit).

The user is requested to enter the total memory size (in bytes) and the number of processes. Then, the program randomizes the arrival time (between 0 and 20) and lifetime (between 10 and 20) of every process. It also randomizes the memory requirements so that the total memory required by all processes does not exceed the total memory size. Then, the program asks the user to choose the algorithm. Memory is then allocated to processes using the algorithm of choice. Note that when a process terminates, its allocated memory becomes free. With every allocation/deallocation operation, your program must print the list of memory holes and the base/limit values of the allocated memory. If not hole is big enough to accommodate the current memory request, your program should report this case showing the total size of available memory.

Here is a sample run:

```
INPUT
Enter the memory size -- 2048
Enter the number of processes -- 3
Generating random timeline
PROCESS | ARRIVAL TIME   |  LIFETIME  | MEMORY REQUIREMENTS
P0          0                  10              512
P1          9                  15              256
P2          20                 17              620
Choose algorithm (BEST[1], WORST[2], FIRST[3]) -- 1

OUTPUT
List of memory holes = [ 2048 ]

Allocating memory to P0 at time 0: [Base = 0, Limit = 512]
List of memory holes = [ 1536 ]
```

```
Allocating memory to P1 at time 9: [Base = 512, Limit = 256]
List of memory holes = [ 1280 ]

Deallocating memory of P1 at time 10
List of memory holes = [512 --> 1280]

Allocating memory to P2 at time 20: [Base = 768, Limit = 620]
List of memory holes = [512 --> 660]

Deallocating memory of P1 at time 24:
List of memory holes = [768 --> 660]

Deallocatig memory of P2 at time 37:
List of memory holes = [ 2048 ]

Done
```