Cairo University
Faculty of Engineering
Computer Engineering Department

OS Memory

Document Structure

- System Description
- Guidelines
- Deliverables

System Description

In this phase, we will drop the assumption that our system has infinite memory. You are required to edit the *Scheduler* (in code file *scheduler.c*) from the "OS Scheduler" assignment (we may call it the *Core* from now on) to include memory allocation capabilities using the following memory allocation policies:

1. First Fit
2. Next Fit
3. Best Fit
4. Buddy System Allocation

Your system should allocate memory space for processes as they enter the system and free it as they leave so that it can be re-used by later processes.

You may make the following assumptions:

1. The total memory size is 1024 bytes.
2. Each process size is less than or equal 256 bytes.
3. Assume single uni-core CPU.

4. The memory allocated for a process as it enters the system is constant over the time it spends in the system and should never change.

5. In **Next Fit,** if the new process to be allocated can not fit after the last allocated process (e.g. the last process was allocated near the end of the memory space, and there is insufficient memory space for the new process), then you should wrap around and start from the beginning.

6. In **Buddy System Allocation,** if there are two memory fragments that can fit an allocation request, then the buddy algorithm will allocate the process in the smaller fragment.

7. Before you **fork** any process in the scheduler as it arrives, you have to check first whether it can be allocated or not.

    a) If there is enough memory space for this process, then the memory is allocated and the process is forked and scheduled.

    b) If there is not enough memory space for this process, then it should be pushed to a waiting list until another process finishes and its memory is deallocated from the system. You should then check the waiting list again if there are any process(es) that can be allocated.

    c) If a process is deallocated and there are two processes in the waiting list:

        i. Allocate both processes if it is possible.

        ii. If one process only can be allocated, then you allocate the oldest.

8. Waiting time is the time that the process has been waiting SINCE its arrival. So, all the clock cycles where the process kept waiting for a memory space in the waiting list will be counted as **waiting time** in addition to the **waiting time** after scheduling when the process is placed in the ready queue due to preemption.

9. The process should be deleted from memory only when it completely finishes its job. The memory should not be released during preemption or during context switching between processes if the process has not still finished its task. (i.e. the process stays allocated in the memory even if it is not running right now (due to a RR scheduling for example)).

Input

You will need to (slightly) modify the the *Process Generator* (in code file *process_generator.c*) to accept an extra process information; *memsize*.

processes.txt example

#id	arrival	runtime	priority	memsize
1	1	6	5	200
2	3	3	3	170

- Comments are added as lines beginning with # and should be ignored.
- Each non-comment line represents a process.
- Fields are separated with *one tab character '\t'*.
- You can assume that processes are sorted by their arrival time. *Take care that 2 or more processes may arrive at the same time.*

Output

A new output file *memory.log* should be generated for the memory information.

memory.log example (Buddy)

#At	time	x	allocated	y	bytes	for process	z	from	i	to	j
At	time	1	allocated	200	bytes	for process	1	from	0	to	255
At	time	3	allocated	170	bytes	for process	2	from	256	to	511
At	time	6	freed	170	bytes	for process	2	from	256	to	511
At	time	10	freed	200	bytes	for process	1	from	0	to	255

- Comments are added as lines beginning with # and should be ignored.
- Make sure both *allocated* and *freed* memory information are logged.
- You need to stick to the given format because files are compared automatically.

Guidelines

- Read the document carefully. You can specify any other additional input to algorithms or any assumption but after taking permission from your TA.
- The running command is expected to include the memory allocation policy as follows:

  ./scheduler.o testcase.txt -sch 5 -q 2 -mem 3

  where mem is the memory allocation algorithm number as specified in Page 1. The remaining parameters are the same as specified in the scheduler assignment.
- Your program must not crash.
- You need to release all the IPC resources upon exit.
- The measuring unit of time is 1 sec, there are no fractions, so no process will run for 1.5 second or 2.3 seconds. Only integer values are allowed.
- You can use any IDE (VSCode, Eclipse, CodeBlocks, 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.
- Spend a good time in design and it will make your life much easier in implementation.
- The code should be clearly commented and the variables names should be indicative.

Deliverables

You should deliver the modified code from the "OS Scheduler" assignment along with a modified version from the report that includes any necessary information about the memory allocation system described in this assignment.