2 - Test Cases

Team:

AJ Omartian - Project Manager

Danya Alrawi and Ikbel Amri - Architect

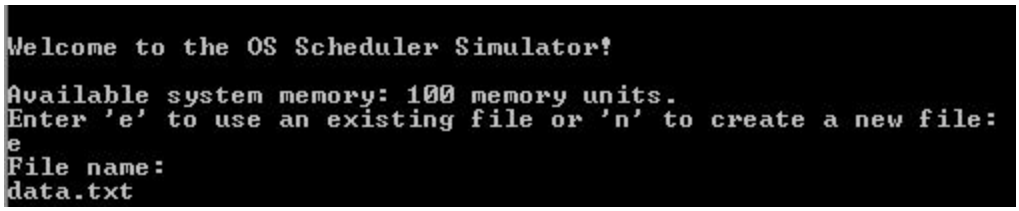
Elisabeth Fernandez - Business Analyst

Alex Hernandez - Lead Software Developer

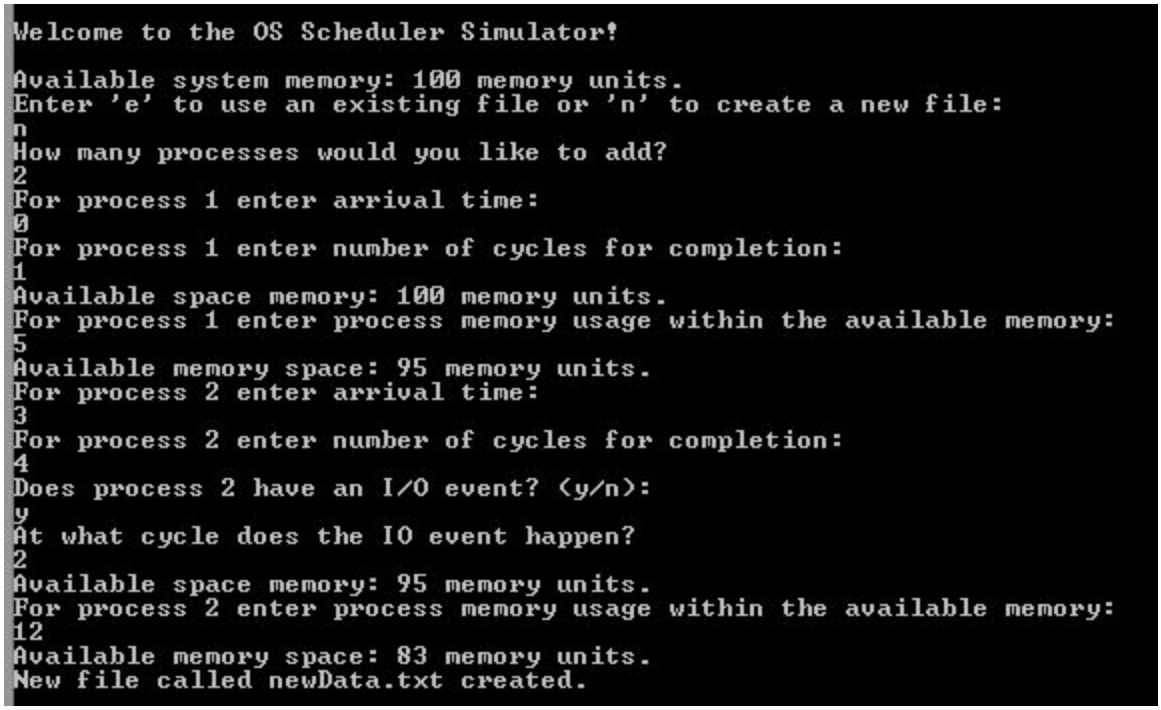
Reading Process List:

The program reads a list of process definitions inputted from either:

- An existing file



- A list of processes inputted in the user interface



We can see that the user is not asked to specify an IO event for the first process because it takes one cycle to complete. This was a design decision we agreed on, that processes with one cycle cannot have an IO event.

Verifying Data:

Verifying the validity of input was a crucial requirement for the functioning of the simulator. For that purpose, we prompted the user to re enter the input values if they did not meet the required specifications. Example:

```
For process 1 enter arrival time:
-5
Invalid input. Arrival time cannot be negative.
Please enter an integer greater than or equal to zero.
0
For process 1 enter number of cycles for completion:
```

Displaying, Adding, Deleting and Making Updates to Process List:

- Displaying the list of active processes: The user has the ability to display the list of active processes by pressing (1) from the main menu after inputting the list:

```
P ID: 1 arrives at T=0 and requires 1 cycles and has no IO event and takes up 5
memory units
P ID: 2 arrives at T=3 and requires 4 cycles and has an IO event at cycle 2 and
takes up 12 memory units
Available memory space: 83 memory units.
```

- Adding processes: The user has the ability to add additional processes to the list of active processes by pressing (2) from the main menu:

```
How many processes would you like to add?
1
For process 1 enter arrival time:
5
For process 1 enter number of cycles for completion:
2
Does process 1 have an I/O event? (y/n):
n
Available memory space: 83 memory units.
For process 1 enter process memory usage within the available memory:
1
Available memory space: 82 memory units.
```

- Deleting processes: The user has the ability to delete processes from the list of active processes by pressing (4) from the main menu. They will be shown a the list of active

processes, asked to enter the ID of the process to delete, then asked if they would like to delete other processes:

```
P ID: 1 arrives at T=0 and requires 1 cycles and has no IO event and takes up 5
memory units
P ID: 2 arrives at T=3 and requires 4 cycles and has an IO event at cycle 2 and
takes up 12 memory units
P ID: 3 arrives at T=5 and requires 2 cycles and has no IO event and takes up 1
memory units
Select the ID of the process you would like to delete.
2
The following process was deleted:

P ID: 2 arrives at T=3 and requires 4 cycles and has an IO event at cycle 2 and
takes up 12 memory units

Would you like to delete another process? (y/n)
n
```

- Updating the active list of processes: The user can update any process from the active list of processes by pressing (3) through a thorough menu. Example of updating the IO event:

```
Select the ID of the process you would like to update
3

P ID: 3 arrives at T=5 and requires 2 cycles and has no IO event and takes up 1
memory units
(a) To modify arrival time
(c) to modify the number of cycles
(i) to modify IO event
(m) to modify the process memory
to return to main menu type 'return'
i
Currently, the process has no IO event. Would you like to add an IO event for Pr
ocess with ID 3? (y/n)
y
Enter clock cycle at which IO event happens
1
IO event set to cycle 1
```

First Come First Serve Algorithm and Memory Management:

- FCFS is a non-preemptive algorithm, therefore processes do not exit until they are complete (i.e. until they run out of clock cycles)

```
Enter dispatch value:
2

Available memory space: 82 memory units.

T0 Process 1 is running.
Process 1 is complete.
5 memory units freed.
Available memory space: 87 memory units.
T 1 Dispatch
T 2 Dispatch

T3 Process 3 is running.
T4 Process 3 is running.
Process 3 is complete.
1 memory units freed.
Available memory space: 88 memory units.
T 5 Dispatch
T 6 Dispatch
```

Round Robin Algorithm and Memory Management:

- This algorithm is pre-emptive, a process it taken out once the time quantum is reached, or the process has an IO event, in which case it is taken out and the next process becomes active.
- Time Quantum: The time quantum value was defined as part of the setup of the environment.
- Dispatch Penalty: The user has the ability to set the Dispatch penalty which is the time penalty for the Dispatcher to update the running process.

```
Enter time quantum value:
2
Processes List:
P ID: 1 arrives at T=0 and requires 1 cycles and has no IO event
P ID: 3 arrives at T=5 and requires 2 cycles and has an IO event at cycle 1


Round Robin (Quantum 2)
T-1 Dispatch Penalty
 P ID: 1 is now active.

Available memory space: 82 memory units.
T0 P ID: 1 is running.
 P ID: 1 is complete
5 memory units freed.
Available memory space: 87 memory units.
T1 Dispatch Penalty
 P ID: 3 is now active.

Available memory space: 87 memory units.
T2 I/O...
T2 P ID: 3 is running.
T3 P ID: 3 is running.
 P ID: 3 is complete
1 memory units freed.
Available memory space: 88 memory units.
```