SYSC4001 Assignment 2 Report
Section L3 - Group 2

Dante Gyetko (101296637)
Vlad Myrutenko (101301663)
Github Repo P2: https://github.com/VladMyrutenko/SYSC4001_A2_P2
Github Repo P3: https://github.com/DanteGyetko/SYSC4001_A2_P3

TRACE 1

We start with only the init process in partition 6.

Fork() is called, which creates a clone of init and places it in the next smallest available partition, which is 5. We then go into the child process' execution, which is an Exec(program1) call. This replaces the child's memory image with that of program1, which is of size 10, meaning it fits/is moved to partition 4. Program1 executes as it would in Assignment 1.

Then, the child's execution ends and the parent process (init) continues, running Exec(program2), replacing its image with program2 of size 15, which is placed in partition 3 (best fit). Again, program2 runs as it would in Assignment 1, and then the simulation ends as we've reached the end of the input trace.

TRACE 2

Note: program numbers were altered from the provided files. 1 → 3, 2 → 4.

We start with only the init process in partition 6.

Fork() is called, creating a clone child of init in partition 5. This child calls Exec(program3), replacing its image to one of size 10 and therefore moving to partition 4. Program3 executes, calling Fork() to clone a child into partition 3 (best fit). Nothing occurs in the conditionals (IF_CHILD, IF_PARENT). The new child of program3 moves on to call Exec(program4), replacing its image with one of size 15, thus remaining in partition 3 (best fit). This program4 runs as it would in A1, reaching the end of its trace and terminating.

Then, the parent program3 (child of init), which was waiting for its own child to reach the end of its trace, also runs Exec(program4).

→ "Exec(program4)" was run twice, outside of the conditional, because neither the parent or child versions of program3 had an EXEC call inside their conditionals.

Lastly, there is a CPU burst at the end of the main trace, which is run by the init process.

TRACE 3

Note: program numbers were altered from the provided files. 1 → 5.

We start with only the init process in partition 6.

Fork() is called, creating a clone child of init in partition 5. There is nothing in the child's conditional branch, so it moves on to the end of the main trace and runs a CPU burst of 10. Then, the parent (init) runs Exec(program5), replacing its image with one of size 10 and moving to partition 4 (best fit). Program5 runs as it would in A1, and then the simulation terminates.

TRACE 4

We start with only the init process in partition 6.

Fork() is called, creating a clone child of init in partition 5. The child calls Exec(program5), which is of size 15, so its it placed in partition 3 (best fit). Program5 runs as it would in A1.

Next, the parent (init) runs a CPU burst of 250 then calls Exec(program1). Here, program1 was defined as size 40, meaning init is replaced with an image of 40, fitting into partition 1. Program1 runs as it would in A1, and the simulation terminates.

TRACE 5

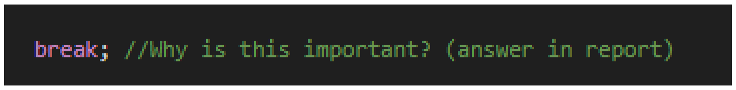
Note: there is a CPU burst at the end of the main trace. This will not get run, and was placed here purely for the purpose of simulation analysis (explained below).

We start with only the init process in partition 6.

Fork() is called, creating a clone child of init in partition 5. The child calls Exec(program4), replacing its memory image with one of size 5, which fits into partition 5 (best fit). Program4 executes as it would in A1.

The parent (init) then calls Exec(program1), replacing its image with one of size 25, fitting into partition 2 (best fit). Program1 executes as it would in A1.

There is a CPU burst of 1000 at the bottom of the main trace. As both the parent process (init) and its child process have called EXEC, replacing themselves with different 'code', *neither actually sees* this CPU burst, and so the simulator (correctly) never shows it running, as can be observed in *execution_5.txt*.



```
break; //Why is this important? (answer in report)
```

This was placed at the bottom of the EXEC activity code.

Because we're iterating through a trace file with a for loop, this is necessary because EXEC *replaces a process trace/image* with a new one. Therefore, it no longer makes sense for the simulator to have the process continue running through the original trace (it no longer exists according to the process that called EXEC).