OS Project1 Report

b07902009 資工二 尚沂瑾

Design

main.c

- Read the input from standard input.
- Call a function `schedule()`. Pass the scheduling rule(FIFO/RR...), process number, and the info of every processes to the function.

scheduler.c

- There are two functions, `schedule()` and `next_proc()`, in this c file.
- `schedule()`: called by main.c
  - Assign cpu 0 to this scheduling process.
  - Sort all child processes by their ready time.
  - Check whether there exists a process whose ready time is equal to the current time. If there exists such process, fork an child process.
  - After forking the child process, block it and wake it up when it is scheduled.
  - Call the function `next_proce()` to find next process for execution.
- `next_proc()`: called by `schedule()`
  - Find next process for execution by the scheduling rule, and return the index of the process.

process.c

- There are four functions, `use_cpu()`, `wake_proc`, `block_proc()`, and `exec_proc()`, in this c file.
- `use_cpu()`:
  - (My system has two cores.)
  - Use sched_setaffinity to assign a task to a specific core.
- `wake_proc`
  - Wake the child process for executing when it is scheduled.
- `block_proc`
  - Block the child process from executing until it is scheduled after it is forked.
- `exec_proc()`
  - Fork a new process for executing the ready process.
  - Assign cpu 1 to the child process.

info.h

- Declare all the functions which are called by different c file.
- Declare a unit of time.
- Declare a process structure to store all the information about a process (pid, ready time, execution time).

Kernel Version

The output of `uname -a`:
`Linux michelle-virtual-machine 4.14.25 #2 SMP Tue Apr 28 11:56:46 CST 2020 x86_64 x86_64 x86_64 GNU/Linux`

Difference of Actual and Theoretical Results

Compare

- Test case 1:
  - Data:

```
FIFO
5
P1 0 500
P2 0 500
P3 0 500
P4 0 500
P5 0 500
```

  - Result:

```
start time of P1: 0 unit
end time of P1: 548 unit
start time of P2: 0 unit
end time of P2: 1074 unit
start time of P3: 0 unit
end time of P3: 1586 unit
start time of P4: 0 unit
end time of P4: 2104 unit
start time of P5: 11 unit
end time of P5: 2613 unit
```

  -
- Test case 2:
  - Data:

```
PSJF
5
P1 0 3000
P2 1000 1000
P3 2000 4000
P4 5000 2000
P5 7000 1000
```

- Result:

```
start time of P1: 0 unit
end time of P1: 4313 unit
start time of P2: 1018 unit
end time of P2: 2059 unit
start time of P3: 3441 unit
end time of P3: 13535 unit
start time of P4: 5756 unit
end time of P4: 8712 unit
start time of P5: 8718 unit
end time of P5: 10092 unit
```

- Test case 3:
  - Data:

```
RR
6
P1 1200 5000
P2 2400 4000
P3 3600 3000
P4 4800 7000
P5 5200 6000
P6 5800 5000
```

- Result:

```
start time of P1: 1203 unit
end time of P1: 20578 unit
start time of P2: 2908 unit
end time of P2: 21221 unit
start time of P3: 4566 unit
end time of P3: 20000 unit
start time of P4: 5892 unit
end time of P4: 34595 unit
start time of P5: 6124 unit
end time of P5: 33495 unit
start time of P6: 6944 unit
end time of P6: 31309 unit
```

- Test case 4:
  - Data:

```
SJF
5
P1 0 3000
P2 1000 1000
P3 2000 4000
P4 5000 2000
P5 7000 1000
```

  - Result:

```
start time of P1: 0 unit
end time of P1: 3455 unit
start time of P2: 2455 unit
end time of P2: 4649 unit
start time of P3: 3455 unit
end time of P3: 9112 unit
start time of P4: 7169 unit
end time of P4: 12204 unit
start time of P5: 9113 unit
end time of P5: 10168 unit
```

Analysis

- Except scheduling, scheduler has many other tasks to do, such as setting cpu affinity, dealing with data structure, doing multiple operations, initializing data, etc.
- For every call of UNIT_TIME(), the actual running time of the function are different.
- There are other tasks on my computer, our process would be interrupted by context switch, however, the clock is still going.