Simulator

Prof. P. Sateesh Kumar

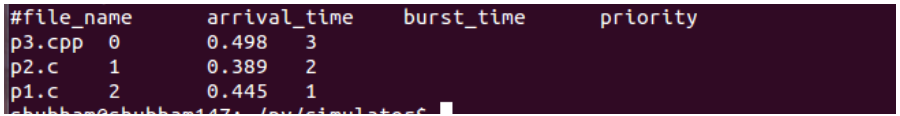
Introduction

The project can be divided into two parts:

User-Interface and The Simulator

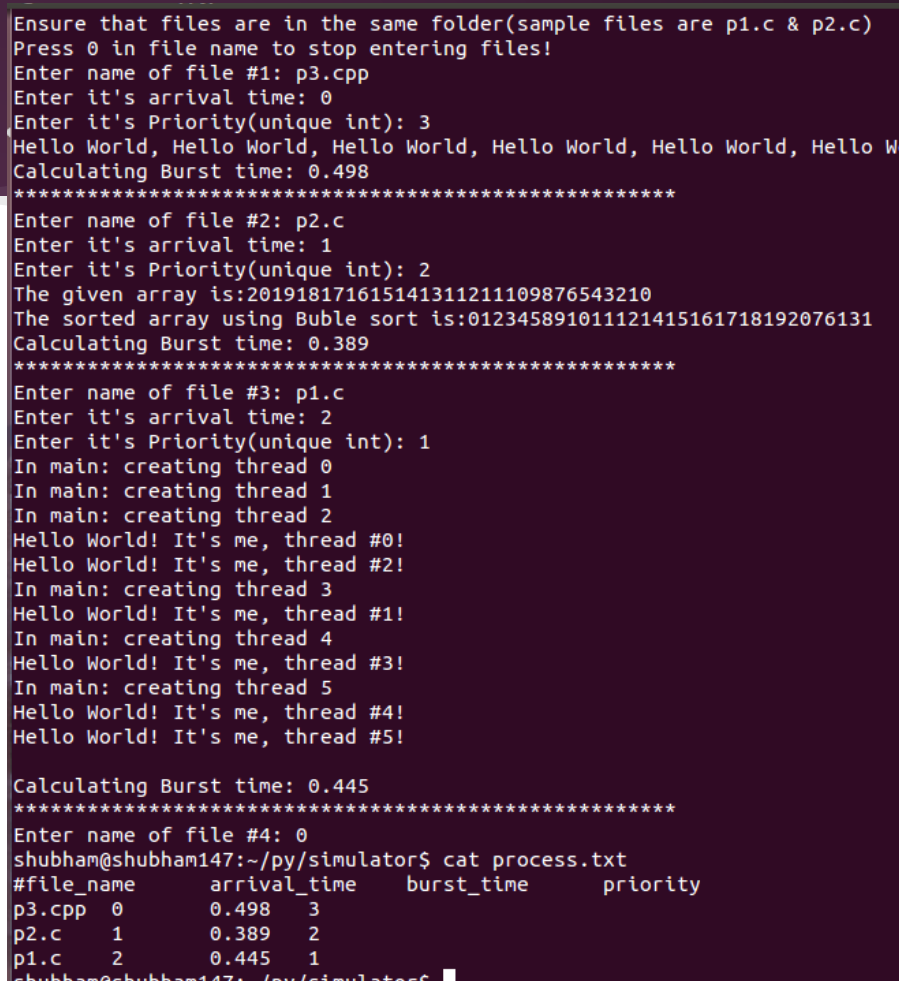
The Interface

- acts as a link between the simulator and the user
- accepts .cpp/.c file as input
- supplies process.txt file as output
- calculates the burst-time and notes the arrival time and priority of the process
- clock variable: used to calculate the burst-time
- the text file thus generated is of the form:



The Interface

- Here we present a screenshot of the user Interface->
- 3 files are run in the interface and they are added to the file process.txt respectively.
- Their burst times are calculated after running them then and there.
- The entering of files is terminated by a ‘0’ in the file name (here, file #4)
- Further, as the burst times are small and real, we will change them to distinct integers using double to _int_ casting during extracting data from file.



Demonstration

FCFS

Name	Arrival Time	Burst Time
P1	0	24
P2	0	3
P3	0	3

0 - 24 P1	24 - 27 P2	27 - 30 P3

```
1
First Come First Serve Scheduling started...
Process P1 Turnaround time is 24.000000, Waiting time is 0.000000
Terminated at 24.000000
Process P2 Turnaround time is 27.000000, Waiting time is 24.000000
Terminated at 27.000000
Process P3 Turnaround time is 30.000000, Waiting time is 27.000000
Terminated at 30.000000

Avg turnaround_time is 27.000000 and avg waiting_time is 17.000000
```

- Average waiting time: (0 + 24 + 27)/3 = 17
- Average turnaround time: (24 + 27 + 30)/3 = 27

SJF

Name	Arrival Time	Burst Time
P1	0	6
P2	0	4
P3	0	1
P4	0	5

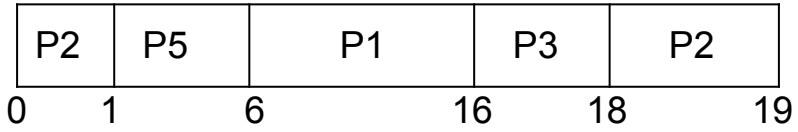
```
2
Non-Preemptive SJF Scheduling started...
Process P3 Turnaround time is 1.000000, Waiting time is 0.000000
Terminated at 1.000000
Process P2 Turnaround time is 5.000000, Waiting time is 1.000000
Terminated at 5.000000
Process P4 Turnaround time is 10.000000, Waiting time is 5.000000
Terminated at 10.000000
Process P1 Turnaround time is 16.000000, Waiting time is 10.000000
Terminated at 16.000000

Avg turnaround_time is 8.000000 and avg waiting_time is 4.000000
```

0-1 P3	1-5 P2	5-10 P4	10-16 P1

- Average waiting time: (0 + 1 + 5+ 10) / 4 = 4
- Average turnaround time: (1 + 5 + 10 + 16) / 4 = 8

Priority scheduling

Name	Arrival Time	Burst Time	Priority
P1	0	10	3
P2	0	1	1
P3	0	2	4
P4	0	1	5
P5	0	5	2




```
3
Priority Scheduling started...
Process P2 Turnaround time is 1.000000, Waiting time is 0.000000
Terminated at 1.000000
Process p5 Turnaround time is 6.000000, Waiting time is 1.000000
Terminated at 6.000000
Process P1 Turnaround time is 16.000000, Waiting time is 6.000000
Terminated at 16.000000
Process P3 Turnaround time is 18.000000, Waiting time is 16.000000
Terminated at 18.000000
Process P4 Turnaround time is 19.000000, Waiting time is 18.000000
Terminated at 19.000000

Avg turnaround_time is 12.000000 and avg waiting_time is 8.200000
```

- Average waiting time: (0+1+6+16+18) / 5 = 8.2
- Average turnaround time: (1+6+16+18+19) / 5 = 12

Round-Robin

Name	Arrival Time	Burst Time
P1	0	24
P2	0	3
P3	0	3

```
4
Enter Quantum Value :4
RR started...
Process P2 Turnaround time is 7.000000, Waiting time is 4.000000
Terminated at 7.000000
Process P3 Turnaround time is 10.000000, Waiting time is 7.000000
Terminated at 10.000000
Process P1 Turnaround time is 30.000000, Waiting time is 6.000000
Terminated at 30.000000

Avg turnaround_time is 15.666667 and avg waiting_time is 5.666667
```

0-4 P1	4-7 P2	7-10 P3	10-14 P1	14-18 P1	18-22 P1	22-26 P1	26-30 P1

- Average waiting time: (6+4+7)/3 = 5.66
- Average turnaround time: (30+7+10)/3 = 15.66

SRTF / SJF Preemptive

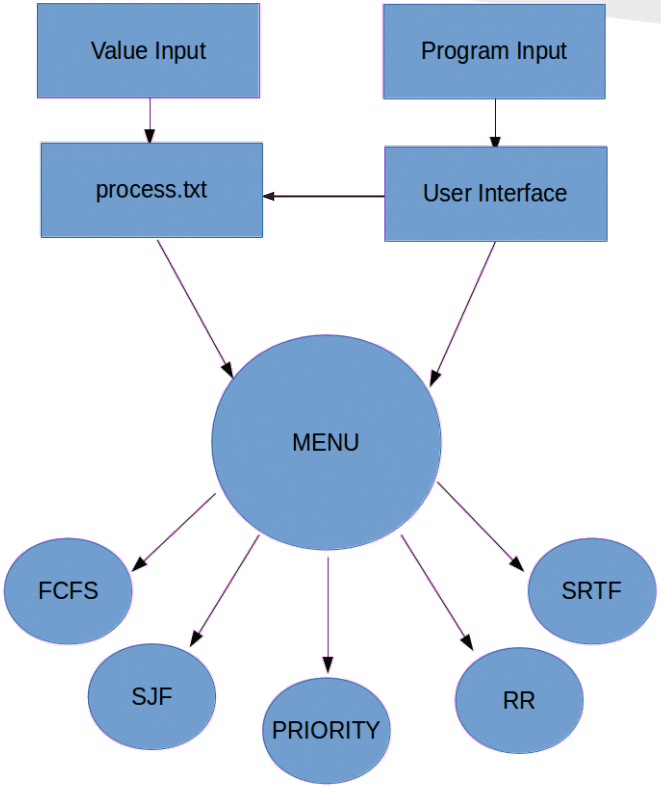
Name	Arrival Time	Burst Time
P1	0	7
P2	2	4
P3	4	1
P4	5	4

```
5
SRTF started...
Process P3 Turnaround time is 1.000000, Waiting time is 0.000000
Terminated at 5.000000
Process P2 Turnaround time is 5.000000, Waiting time is 1.000000
Terminated at 7.000000
Process P4 Turnaround time is 6.000000, Waiting time is 2.000000
Terminated at 11.000000
Process P1 Turnaround time is 16.000000, Waiting time is 9.000000
Terminated at 16.000000

Avg turnaround_time is 7.000000 and avg waiting_time is 3.000000
```

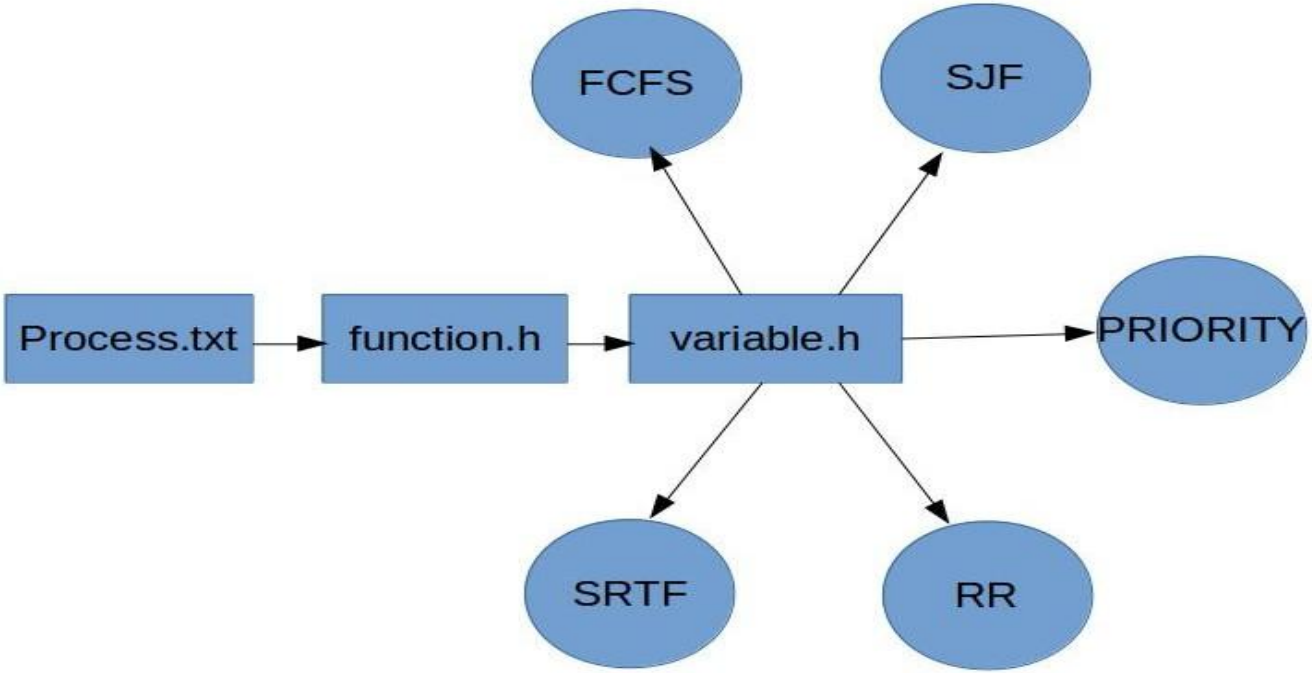
0-2 P1	2-4 P2	4-5 P3	5-7 P2	7-11 P4	11-16 P1

- Average waiting time: (9+1+0+2) / 4 = 3
- Average turnaround time: (1+5+6+16) / 4 = 7

General Layout



General Layout



Variables Defined

- All variables are defined in **variable.h** module.

- Main variables used are :
  - incoming_process
  - total_process
  - process[]
  - _data_
  - quantum

Parsing Input

- Functions used for reading input from **process.txt** are written in module **function.h**

- Main functions used are:
  - get_detail_from_file()
  - split()
  - strtok() from string.h

Pthread

Pthreads library in C provides an **API for creating and managing threads** in any program. It has *three* parts:

1. Initialization
2. Creation and passing relevant Parameters
3. Termination and Joining

Thread identifier and attributes

pthread_t tid;

pthread_attr_init (&attr);

1. declare identifier for thread we will create.

2. this function call is used to set thread attributes.
   a. we do not need to set attributes explicitly
   b. we use default attributes

Creating pthread

➔ *thread_con = pthread_create(&thread, NULL, &(fcfs_thread), &data);*

- **pthread_t** is an type which acts as a handle for the new thread
- **pthread_attr_t** is another data type which allows you to fine tune various parameters, to use the defaults pass NULL.
- **thread_function** is the function the new thread is executing, the thread will terminate when this function terminates, or it is explicitly killed
- **arguments** is a void * pointer which is passed as the only argument to the thread_function.
- **thread_con** is ‘zero’ if thread is successfully created.

Termination and Joining

pthread_exit(0)

- The thread will complete its execution when the above function is called.

pthread_join(tid,NULL);

- The thread which encounters this statement will wait for the (child) process to complete execution with the id “tid” and commence its execution after it happens.

FCFS

- ➔ processes are ordered according to their arrival time
- ➔ inside function **FCFS,** from the array of processes, determine the process with shortest arrival time
- ➔ create pthread using **data** of given process as argument to a function (**fcfs_thread**)
- ➔ modify various parameters of the process and algorithm inside the function
- ➔ after joining child process with parent process, update total waiting and turn-around time
- ➔ repeat above 4 steps inside for loop till all processes have been processed

Disadvantage of FCFS :

The average waiting time for FCFS is usually quite long and there is a possible convoy effect in which all processes wait for one big process to get off the CPU.

SJF

1. A function shortest_process is called to determine which of the remaining processes has the least burst time.
2. Using Pthreads the selected process is run and the waiting time and turnaround time for the process is calculated.
3. The child process is terminated and joined with the parent process.
4. The waiting time and turnaround time for other processes and the scheduler is updated.
5. Repeat.

Advantages/Disadvantages of SJF

Advantages:

1. Simplicity.
2. Minimizes waiting time.

Disadvantages:

1. Potential for Process Starvation.
2. Burst time of a process must be known before its execution.

Round Robin(RR) Algorithm

1. Ready Queue is treated as **circular FIFO queue**
2. CPU scheduler goes around ready queue, allocates CPU to each process for time interval up to one quantum
3. process having remaining time less than quantum leaves the CPU voluntarily
4. otherwise, context switch is executed
5. Rule of thumb: 80% of CPU bursts should be shorter than the time quantum.

Implementation in Project

- two functions:
  - **void rr(void)**: main function (thread)
  - **void *rr_thread(void *)**: used in pthread_create
- **pthread_create** for each quantum

Implementation in Project

- Steps:

1. calculate total burst time
2. create three arrays **thread_con**, **thread**, **data**
3. create new thread for each quantum
4. in each thread
   if (**process[curr_index].remaining_time <= quantum**),
   then we terminate the process and calculates TT, WT
   else
   process[curr_index].remaining_time- = quantum
5. calculate AWT ATT

Priority scheduling

- A priority(integer no.) is associated to each process, and the CPU is allocated to the process with **highest** priority(1=highest priority).
- It can be preemptive or non-preemptive.
- Equal priority process are scheduled in FCFS order.
- Major problem with this scheduling is **starvation** or indefinite blocking.
- Solution to this is **Aging**.

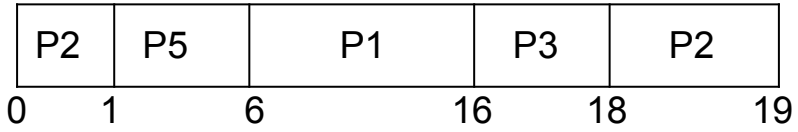
Priority scheduling

Implementation:

- void priority_scheduling( ) - initiates the scheduling algorithm and calls the function void *priority_scheduling_func( )
- int priority_order( ) - the function dynamically enqueues the process in their priority order
- void *priority_scheduling_func( ) - the function implements the priority scheduling algorithm
- calculated avg.turn around time and waiting time of given processes
- Aging is to be implemented

Priority scheduling

Name	Arrival Time	Burst Time	Priority
P1	0	10	3
P2	0	1	1
P3	0	2	4
P4	0	1	5
P5	0	5	2




```
3
Priority Scheduling started...
Process P2 Turnaround time is 1.000000, Waiting time is 0.000000
Terminated at 1.000000
Process p5 Turnaround time is 6.000000, Waiting time is 1.000000
Terminated at 6.000000
Process P1 Turnaround time is 16.000000, Waiting time is 6.000000
Terminated at 16.000000
Process P3 Turnaround time is 18.000000, Waiting time is 16.000000
Terminated at 18.000000
Process P4 Turnaround time is 19.000000, Waiting time is 18.000000
Terminated at 19.000000

Avg turnaround_time is 12.000000 and avg waiting_time is 8.200000
```

- Average waiting time: (0+1+6+16+18) / 5 = 8.2
- Average turnaround time: (1+6+16+18+19) / 5 = 12

SRTF SCHEDULING

Following functions have been implemented
1.void srtf(void)--->This function calls pthread_create func., prints the avg. turn around time,burst time of the input processes.
2.int srtf_shortest(float)---> Invoked inside the previous function,returns the process with shortest remaining time.
3.void *srtf_thread(void *)-->Calculates and prints the turnaround, waiting and terminating time of individual child processes.

**The advantage of SRTF is
Short processes returns quickly and it has high yield (jobs done per minutes).
**The Disadvantages of SRTF are:-
It must be possible to accurately estimate a job's remaining time
Large jobs may never get their time share if many small process is continually added (process starvation).

OVERVIEW & MODIFICATIONS

We have

1.Successfully used the User-Interface to calc Burst time.

2.Next we successfully implemented the Simulator interlaced with the UI.

3.5 Scheduling algorithms were implemented on the processes.

4.P-Threads were implemented in creating the child processes.

5.Avg. Waiting and Turn Around time calculated.

We plan to:

1.Append the priority and SRTF algo. with aging

2.Add Multilevel Queue Scheduling Algo

to the present 5

3.Implement the algorithm for various Memory Management techniques.