Documentation
OS_ Assignment_3

Git-Hub :- https://github.com/A-WASIF/OS

Explanation:-
This program allows users to submit commands with optional priorities. This program supports four priority levels (1-4) and manages the execution on the basis for their priorities. After giving commands, the process once created should not directly start the execution of the a.out. It would wait for some signal from the SimpleScheduler.

Major Components and Functions in the given code listed here:

a)Header files are:
1.<stdio.h>: For standard input and output.

2.<stdlib.h>: General utilities libraries.

3.<signal.h>: For signal handling functions.

4.<string.h>:for string manipulation.

5.<unistd.h>:POSIX operating system like :(fork,sleep).

6.<errno.h>:for Error number

7.<sys/types.h>:data types used in system calls.

8.<sys/wait.h>:process control and wait.

9.<sys/time.h>:for time related .

10.<stdbool.h>: Boolean data types and values .

11.<ctype.h>:character handlings .

12.<sys/mmap.h>:-For memory mapping

13.<sys/stat.h>: File status functions.

14.<fnctl.h>:file control functions .

15.<semaphore.h>: Semaphore synchronization functions.

b)Data structures :
1.Queue:- It contains pointers to the front and rear nodes to the queue and represents the priority queues .
2.Qnode:- It represents a node in the priority queues and contains pid,command string ,process state ,priority,execution time , and a pointer to the next node.
3.Histroyentry:- It contains command string,pid ,priority,execution time,and waiting time.
4.shm_t:- It represents the shared memory structure ,containing semaphores and four priority queues .

c)Functions:

1.userinputFunction: It takes input from the user and returns a string .

2.createQueueFunction: Creates an empty priorities queue and returns a pointer to it.

3.newnodeFunction: creates a new node for the priority queues .

4.enqueueFunction: Enqueues a new process node into the specified priority queues ..

5.DequeueFunction: Dequeues and returns a process node from the specified priority queue.

6.isemptyFunction: check if specified priority is empty.

7.printqueueFunction: prints the element of the specified queue .

8. checkforpriorityFunction: check if the user contains a valid priority level .

9.signalhandlerFunction: For handling sigint signals .

10.launchFunction: launch a new process with the given command using execlp.

11.printallqueueFunction: Print all the priority queues.

12.min(): returns minimum of two integers .

13.schedulerFunction: implements the process scheduler logic based on the specified number of processes ,cpus,and time slice .SimpleScheduler maintains a ready queue of processes that are ready for execution. Processes are
added to the rear of the queue, and taken out from the front of the queue in a round-robin fashion.SimpleScheduler’s job is really simple. It should simply take NCPU number of processes from the front
of the ready queue and signal them to start running. After the timer expires for TSLICE, SimpleScheduler will signal the ncpu running processes to stop their execution and add them to the
rear of the ready queue.
14.InitiliazewaittimeFunction: Initialize the waiting time for a process with a specific priority level .

15.callwaittimeFunction: calculated waiting time for a process.

16.addtohistoryFunction: Adds a new entry to the command history .

17.printhiostoryFunction: prints the command history including the command ,execution time and waiting time.
18.checkforpriorityFunction:It is responsible for which process has the highest priority.
19.InitialiseWaitTimeFunction:It calculates the waiting time for a processor which is calculated by the previous number of processes multiplied by tslice.
20.PrepAndLaunchSchedularFunction:It prints the number of each process and then calls the waiting time function and after that calls the scheduler function for execution.

18.MainFunction:1.Parses the command line arguments (number of cpus and time slice) and initializes shared memory and semaphores .

2.Reads the user input ,processes submitted commands ,and manages the execution of processes based on priorities .
3.Implements the scheduling logic for each priority level ,updating waiting times and executing processes accordingly .
4.Prints the command history and cleans up shared memory resources before exiting.

Detailing SimpleScheduler implementation:

A simple scheduler based on the priority levels and a round robin like approach to ensure fairness.
1.First of all ,commands are categorized into four priority levels q1,q2,q3,q4 and each priority level represents a queue of processing, waiting to be executed and higher priority levels have precedence over lower ones ensures the higher priority processes are executed first .

2.The scheduler works in round robin ,selecting the process from the queues for execution and at each round,a fixed number of processes (NCPU)are selected for execution.

3.Scheduler uses fork() system call to create child processes for the selected process and child processes are started (SIGCONT) allowing them to execute for a fixed time slice(TSLICE).

4.After the time slice ,child processes are paused to allow other processes to run.

5.After each time slice ,processes are checked for completion or the need to change priority and if a process has not completed its execution time ,it is moved to a higher priority queue.

6.This Rotation ensures that processes in lower priority are eventually moved to higher priority queues ,preventing starvation.

7.After all the above steps ,now waiting time for process is calculated on the number of time slices they spend waiting in the queues .Waiting time represents the total time a process has been in the system without execution .Waitin time is updated for process in the queues after each round of scheduling .

8.If a process's priority is increased due to waiting ,it is moved to a higher priority queue and new tasks submitted with higher priority are added directly to the corresponding higher priority queue .Processes who complete their execution are removed from the queues.If a process’s priority is increased due to waiting ,it is moved to a higher -priority queue.

9.Shared memory and semaphores are used for synchronizing access to the shared data among multiple process .Signal handling is utilized to control the execution and pausing of tasks .Scheduler ensures efficient utilization of resources by allowing process to execute in parallel ,up to the number of cpus.

Contributions :

Wasif:

Constructtheschedulerportion,signalhandler,launch,initilaisewaitime,calling waittime,preandlaunchshcedular, Create a shared memory in scheduler ,priority.

Vipul Verma:

createqueue,dequeue,printqueue,command line argument,addtohistory,printhistory,documentation,error check.