Project Documentation

Table of Contents

1.Introduction

Project title: Simulating operating system's scheduling protocols with Python

Data: 4/7/2024

Course: Operating system

2.Project Overview

This project aims to simulate the ways of operating system scheduling's protocols

- First Come First Serve (FCFS)
- Round Rubin (RR)
- Shortest Job First (SJF)
- Shortes Remaining First (SRF)
- Earliest Deadline First (EDF)

each protocol has its own way to schedule processes, which process have to be ran, and which process have to be interrupted and sent to ready queue to be ran again later.

I used Python and its oop (Object-Oriented Programming) to write a clean code to study & understand scheduling's protocols.

About each protocol:

1. FCFS (nonpreemptive)

First process come will be running until it finishes.

2. RR (preemptive)

Each process has quantum time if it doesn't finish yet, it be interrupted it and sent to ready queue to be ran again later, until it finishes'

3. SJF (nonpreemptive)

This protocol has a priority that shortest brust time, it will be entered running queue first, until it finishes.

4. SRF (preemptive)

At each process comes the scheduler compare between all ready processes by its remaining time, the shortest will be came first

5. EDF (preemptive)

This protocol is Real-Time that mean each process has deadline if it doesn't finish before it, the process will failed

3.Code Description

I used oop because, it makes development of code comfortable, specially instead of change the same thing at all 5 scheduling, change once at parent class and they will inherit the update.

At each file:

- ### SchedulingCLS.py

This is the parent class that each scheduling class should inherit from it, because it has methods make implementation easier.

Import ABC, abstractmethod to make this class abstract to write abstract method "run( )" that must be implemented at all childes

def __init__(self):

that initiate the all parameters that self "Object" needs it

def finish (self):

that has the sequence when the scheduler finish the process

def unfinish (self, rem):

that has sequence when the process has been interrupted and hasn't finished yet, it take rem "integer" that reference to remaining time

def to_String(self):

that print the Object's variables

def fit (self, process_list):

that pass the processes list to object to schedule the processes

def readyQueue (self):

that watch the processes arrival time and push the process which came to ready queue

def coming_after (self, rem):

that check if there are processes will come during specific time, it take rem "specific time"

def calc (self):

that calc the avg Variables in object

- FCFS_CLS.py, RR_CLS.py, SJF_CLS.py, SRF_CLS.py & EDF_CLS.py

They inherit from parent class, so each class has own implementation's run method.

- BachEND.py

That file response at file handling, fetch data from input file and create output file show scheduling.

Import prettytable lib for make tables easer.

def read (inpFile, lst):

take the input file name, and the variable list which storage processes list

def output (fcfs, rr, sjf, srf, edf):

take 5 scheduling objects to create output file

- Main.py

That file has the main method “while True”, it take the file name and the quantum time for RR scheduling, and call the five scheduling object and run them, finally create output file, then repeat the loop asking for the input file name.

4.Git Hup Link

The all code is here:

sondosamr/Scheduling_Simulater_Python: "This is a university project focusing on CPU scheduling simulation in operating systems, implemented using Python." (github.com)