WORKSHEET

A10.5 Scheduling algorithms

Operating systems, software and translators · A level · OCR H446 1.2.1, AQA 7517 4.6.1.4, Eduqas A500QS 2.6 · about 45 min

BugBotLab

Name _____ Class _____ Date _____

What this lesson is about

First come first served, round robin, shortest job first, shortest remaining time and multi-level feedback queues, compared on one set of processes.

- What a scheduler is trying to do
- One set of processes, five algorithms
- Comparing them

Questions

5 marks in all

1. Which scheduling algorithm is pre-emptive and chooses the process with the least time left to run? [1 mark]

- ☐ **A** Shortest remaining time
- ☐ **B** Shortest job first
- ☐ **C** First come first served
- ☐ **D** Round robin

2. Processes P (arrives 0, burst 6), Q (arrives 1, burst 3) and R (arrives 2, burst 1) are scheduled first come first served. What is the average waiting time? [1 mark]

3. A long process never gets the processor because short processes keep arriving. What is this called, and which algorithm can cause it? [1 mark]

- ☐ **A** Starvation; shortest job first
- ☐ **B** Thrashing; round robin
- ☐ **C** Deadlock; first come first served
- ☐ **D** Starvation; round robin

4. Which statements about multi-level feedback queues are true? [1 mark]

Tick every answer that is true.

- ☐ **A** A process that uses its whole time slice moves to a lower-priority queue
- ☐ **B** It does not need to know burst times in advance
- ☐ **C** A process that has waited a long time can be moved up to prevent starvation
- ☐ **D** Every process stays in the queue it started in

5. What does this program print? [1 mark]

```
queue = [('A', 5), ('B', 2), ('C', 4)]
slice = 3
time = 0
while queue:
    name, left = queue.pop(0)
    run = min(slice, left)
    time = time + run
    if left > run:
        queue.append((name, left - run))
    else:
        print(name, 'finishes at', time)
```

The task: waiting times

Write `sjf(processes)` and `srt(processes)`. `processes` is a list of tuples `(name, arrival, burst)`: `name` is a string, and `arrival` and `burst` are whole numbers of time units (arrival 0 or more, burst 1 or more). Each function returns the **average waiting time** as a float, where a process's waiting time is its finish time minus its arrival time minus its burst time. - `sjf`: non-pre-emptive. Whenever the processor is free, run the arrived process with the smallest burst time to completion. If nothing has arrived, move the clock on to the next arrival. - `srt`: pre-emptive. Move the clock one unit at a time; each unit, run the arrived, unfinished process with the least time left. - In both, break a tie by choosing the process that arrived first. Keep the `fcfs` line and print all three, to 2 decimal places, one per line: `FCFS average wait: <x>`, `SJF average wait: <x>` and `SRT average wait: <x>`. The robot stays still.

```
# the two lines every program starts with: the commands, then the robot
from bugbot import *
connect()

processes = [("A", 0, 8), ("B", 1, 4), ("C", 2, 9), ("D", 3, 5)]

def fcfs(processes):
    time = 0
    total_wait = 0
    for name, arrival, burst in sorted(processes, key=lambda p: p[1]):
        time = max(time, arrival)
        total_wait = total_wait + time - arrival
        time = time + burst
    return total_wait / len(processes)

print(f"FCFS average wait: {fcfs(processes):.2f}")
```

Plan your program here, then type it in and press Run.

Do it on the robot

www.bugbotlab.com/learn/a10-5-scheduling/

The simulator checks it and tells you when it passes. Nothing to install, no account.

Challenges

1. Write `round_robin(processes, quantum)` and check it gives 13.50 for a slice of 3. Which slice gives the lowest average wait for these processes?
2. Add a fifth process E that arrives at 4 with a burst of 1. Which algorithm helps it most?
3. Show SJF starving a process: invent a stream of short processes, arriving one after another, that keeps C waiting for as long as they keep coming.