CS3104 – Operating Systems

Assignment: P2 – An analysis of schedulers

Deadline: 18 November 2021 *Credits*: 50% of coursework mark

MMS is the definitive source for deadline and credit details

You are expected to have read and understood all the information in this specification and any accompanying documents at least a week before the deadline. You must contact the lecturer regarding any queries well in advance of the deadline.

Aim / Learning objectives

The purpose of this assignment is three-fold:

- to provide insight into a real scheduling algorithm;
- to analyse real-life behaviour of different programs;
- to understand how a program's burst behaviour affects scheduling.

Background

BSD (Berkeley Software Distribution) is a version of UNIX originally developed at UC Berkeley. It has spawned numerous descendants over the years. These remain a popular alternative to operating systems based around GNU and Linux.

In this coursework exercise you will research the traditional BSD scheduler (sometimes called 4.4BSD scheduler, after the corresponding BSD version). You will also profile a set of real programs and analyse the expected behaviour of the scheduler based on these profiles. The practical consists of five parts, with the first three parts corresponding to the basic spec. You are encouraged to tackle them in order because each part builds on previous ones.

Part 1: Research

You should research and write a short summary about the classic BSD scheduler (as used by OpenBSD, NetBSD, and older versions of FreeBSD). Your description should not be longer than **1500 words** (advisory limit). Overlength will not be penalised, but we will reward conciseness and precision and may penalise lack of focus and lack of clarity. In your description, make sure to relate your chosen scheduler to concepts covered in lectures and the textbook such as (but not limited to):

- what class of algorithm the scheduler fits into (e.g. round-robin, feeback queue, FIFO),
- is it pre-emptive or not,
- how well it works in multi-core and multi-CPU environments,
- what data structures are used to represent processes and queues,
- how priorities are represented and handled, etc.

You can use the references provided in this spec as a starting point, but you are encouraged to research beyond this and clearly identify all the sources you used. You are encouraged to make use of diagrams as appropriate.

Part 2: Profiling

You are provided with (Linux-based) starter code which launches an external program and records each time this program enters or leaves a system call. This can be done on the lab machines. There is no need to use a BSD computer for this! The starter code can be found on studres:

http://studres.cs.st-andrews.ac.uk/CS3104/Practicals/P2-SchedulingAnalysis/profile.c

You should modify this program to record (for example by outputting to standard output or writing to a file) all timestamps over the duration of a program's execution -- thus generating a profile which shows which time periods were spent in user mode, and which in kernel mode. In this practical, we

will use the time spent servicing system calls (in kernel mode) to represent I/O bursts, and the time spent in user mode to represent CPU bursts. This is obviously a crude approximation, but it allows us to easily profile different programs.

You will then use your program to profile three different programs of your choice. The programs should be chosen to represent different I/O profiles (e.g. an I/O-driven program that relies on user input or disk access, a CPU-driven program that is computation-driven, and a program that exhibits a combination of heavy I/O and heavy CPU processing). Justify why your choices are relevant and present a small section of each of these profiles in your report. Concentrate on a short period of time (about 10ms) and plot the sequence of I/O and CPU bursts for each of the three programs during this period. Explain any similarity or difference between these three plots -- are they expected or surprising?

Part 3: Analysis

Having researched the algorithm and profiled some processes, you should now explain how your scheduler would schedule these processes. To do this, start with the profiles you created in Part 2. Assume that all three processes arrive at the same time and have the same priority. The scheduler will then need to decide which process to run at what time.

In this part, you will create a short schedule based on your understanding of the scheduler from Part 1. Based on your understanding, show the sequence in which the processes will be scheduled, and (approximately) at what time each process takes control of the CPU. You only need to show the first 10 scheduling decisions, see Fig 1 for a rough example of what this could look like. You will have to explain how you determined this schedule based on your understanding of the algorithm and the profiles you created.

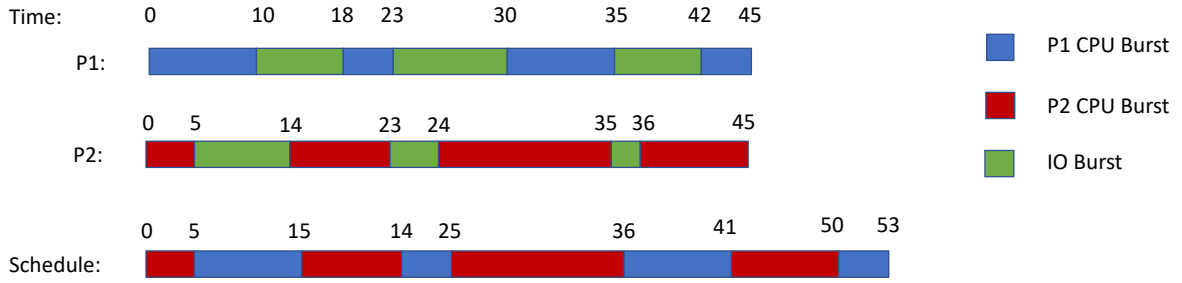

Figure 1: Example Schedule

Part 4: Study of source code (Advanced)

This is an advanced part and should only be attempted once you have successfully completed parts 1-3 to a high standard. In this part, you should have a look at the source code for the BSD scheduler you described in Part 1 and provide a high-level explanation of how it works at the code level. The source code for the OpenBSD scheduler can be found at:

https://github.com/openbsd/src/blob/master/sys/kern/sched_bsd.c

Understanding real-world kernel code can be challenging, so focus on main functionality. Line-by-line explanations are not required. It is more important to relate sections of the code to the main scheduling concepts from the module, such as (but not limited to):

- can you identify how Process Control Blocks (PCBs) are represented and stored,
- can you identify any data structures used to store PCBs,
- can you find how priorities are represented and where in the code they are used,
- can you find where the scheduler decides whether to pre-empt a process, etc.

You might not be able to find all of this easily, but there may be other relevant concepts that you do find.

Part 5: Comparison (Advanced)

Only attempt this part after completing parts 1-4. Many BSD variants have introduced more advanced schedulers in recent years. FreeBSD now defaults to the ULE scheduler, and NetBSD defaults to the M2 scheduler. DragonflyBSD has a two-tiered scheduler with kernelspace and userspace components. Read up on **one** of these more advanced schedulers and explain how it differs from the traditional scheduler you analysed in parts 1-4. What are the advantages and disadvantages of this scheduler compared to the classic BSD scheduler? How does using this scheduler change your analysis in Part 3? What are the main differences in terms of implementation?

Submission

Submit a single ZIP file containing the code you used to generate the profiles in Part 2, and a PDF document containing a report that documents your work on all four parts of the practical. The report should contain a short introduction and conclusion, and one section for each part listed above. Make sure to cite all external material and to number and caption any figures.

Resources

You may find some of these sources a good starting point, but you are encouraged to look for others:

https://github.com/openbsd/src/blob/master/sys/kern/sched_bsd.c
https://manikishan.wordpress.com/2020/05/10/scheduling-in-netbsd-part-1/
https://flylib.com/books/en/2.849.1.57/1/
https://www.informit.com/articles/article.aspx?p=366888&seqNum=4

Assessment
Marking will follow the guidelines given in the school student handbook (see link in next section). Some specific descriptors for this assignment are given below:

Mark range	Descriptor
1 - 6	A submission that shows little evidence of any attempt to complete the work.
7 - 10	A poor description of a real scheduler (Part 1) which is lacking in clarity, detail and understanding, and does not evidence additional research beyond materials covered in the module.
11 - 13	Successful completion of Part 1 and Part 2 which shows a good level of detail and understanding, and manages to effectively relate the scheduler to concepts covered in lectures.
14 - 16	A submission which completes Parts 1-3 to a good standard. The description should be correct, informative, and demonstrate good understanding of lecture concepts and make use of external sources. The profiles should be correctly extracted (as evidenced by the code), well-motivated, and well-explained in the report. Part 3 should show a correct schedule and demonstrate a solid understanding of how the scheduler works in practice.
17-18	A submission which completes Parts 1-4 to an excellent standard. It should show evidence of extensive background reading and understanding of strengths and weaknesses of the presented scheduler. Part 4 should demonstrate good understanding of the scheduler's source code. All work and the report must be excellent for this grade band.
19-20	A submission which completes Parts 1-5 to an exceptional standard with no significant weaknesses.

Policies and Guidelines

Marking
See the standard mark descriptors in the School Student Handbook:
http://info.cs.st-andrews.ac.uk/student-handbook/learning-teaching/feedback.html#Mark_Descriptors

Lateness penalty
The standard penalty for late submission applies (Scheme B: 1 mark per 8 hour period, or part thereof):
http://info.cs.st-andrews.ac.uk/student-handbook/learning-teaching/assessment.html#lateness-penalties

Good academic practice
The University policy on Good Academic Practice applies:
https://info.cs.st-andrews.ac.uk/student-handbook/academic/gap.html