COM1032- Coursework

Description:

This is an individual coursework assignment. You are encouraged to use any of the lecture material and lab exercises, and to complete the assignment you will need to further research relevant material on the internet. Discussion of the coursework is encouraged however, the work that you submit should be your own individual effort.

Release	Friday 1st of March 2019, Week 4
Deadline	Tuesday 26th of March 2019 @ 4pm, Week 8
Weighting	40%

General Aim:

You are required to submit a project that covers the operating system concepts that were covered in COM1032 Mobile Computing Module.

Submit your source code in the source folder as one zip file and a separate report document (as pdf) via SurreyLearn under the assignment section in coursework folder. See later the section in this description for what to submit.

Marking and feedback:

Feedback will be provided within the 3 standard semester weeks. The marks distribution is provided inline with the requirement descriptions below, along with the rubric and mark descriptors at the end.

IMPORTANT: The questions are separated into tasks (*T*). For each one of the tasks give a brief description/explanation/evaluation (approximately 1-3 paragraphs per task) of the design choices/implementation/analysis of performance measures and take a screenshot (or more, if appropriate) to support your answer. Submit all source code written including inline documentation through commenting and applying good programming style as documented in this link:

https://google.github.io/styleguide/javaguide.html

Coursework Tasks:

The coursework tasks are designed to reflect your understanding of the concepts (learning objectives). These are detailed in numbered multilevel list below. The marks distribution shown below is out of 100, but will be weighted to 40% of the module total.

Task 1 (T1): Process Management and scheduling: – 25 marks

The following will require reading through the Minix Processes and Minix Source Code documents in SurreyLearn week 2 other materials. In the Minix Processes document, in System Startup section, you will find that the main.c initialises Minix kernel and manages the execution of the system. You will also find that lines 7172 to 7242 is where the process table is initialised. Tracing through these

lines, and where the variables used are initially declared and initialised, you will be able to answer the following questions. In the Minix Processes document in Process Scheduling section, there is more explanation about how processes are added to ready queues, and which scheduling algorithm is implemented by Minix. Include your answers in your report under each task header.

T1(a): Minix Source Code Understanding - 10 marks

Find out from the Minix source code how the Minix developers have implemented the following:

i. Process Control Block contents and purpose of each element **– 2 marks**
ii. How Minix manages processes State Transitions **– 4 marks**
iii. Explain 2 Kernel processes and their interfaces through system (server) calls and user invocations. **– 4 marks**

T1(b): Process Scheduling Simulator - 15 marks

Download from SurreyLearn in week 2 section the process scheduling simulator (pss.jar). The simulator takes an input file describing the processes to add to the ready queue, and simulate their running using the selected scheduling algorithms, and generate some performance metrics and charts. You are required to configure the input file based on your URN numbers, and run your own simulations. You need to include in your coursework report document the resulting chart with detailed 1:2 paragraph(s) for each algorithm explaining the resulting performance based on your understanding of the input configuration parameters and the selected algorithm. These scheduling algorithms are as follows:

- - First come first serve (FCFS) **– 3 marks**
- - Shortest job first (SJF) **– 3 marks**
- - Round robin (RR) **– 3 marks**
- - Highest Priority First static priority, preemptive (HPFSP) **– 3 marks**
- - Weighted round robin (WRR) **– 3 marks**

To configure the input file, for example if your URN is 6231528:

1. First two lines are for you to write comments
2. Third line is the number of processes. Configure your simulation to run for ***x*** processes such that ***x*** is the right-most digit in your student ID. This would be 8 in this example.
3. For every process from the first to ***x***, include a line in the input file. . If x in your ID is < 4, you should define 4 processes at least in the input file to appreciate the differences of the different scheduling algorithms. This file contains the following items separated by spaces:
   a. Serial number for the process beginning from 1 up to ***x*** in the last line that should be PID (Process ID).
   b. The time the process arrived to the ready queue with accuracy up to one digit after the decimal point. This would be ***a.b***, in which ***a*** is the first number in your URN, and ***b*** is the second for the first process, such as 6.2, then second number as ***a*** and third number as ***b*** for the second process such as 2.3, and so forth until you finish the ***x*** processes or start again from 6.2 in cycles until you finish.
   c. The time the process is expected to run for (CPU Burst): ***y***, in which ***y*** should be the first number in your student ID multiplied by 10: for the **first** process (60 in this example), and **second** number in your student ID to the second process (20 in this

example) and so forth until you reach ***x***, or repeat in cycles again to finish defining priorities to all your ***x*** processes.

d. The process priority ***p***, in which ***p*** should be the **last** number in your student ID for the first process (8 in this example), and **second last** number in your student ID to the second process (2 in this example) and so forth until you reach ***x***, or repeat in cycles again to finish defining priorities to all you ***x*** processes.

4. From the GUI screen choose your quantum value ***q***, such that ***q*** is the middle number if your URN multiplied by 10, in this example is 10.

The graph shows the number of processes on the y-axis. The time line of the processor is shown in the x-axis. A horizontal line in the graph is the set of points from the time the process is ***scheduled*** (added to the ready queue), to the time it is ***either terminated or interrupted***. Each of these lines are horizontal in the graph and corresponding to the process number, and represent a visualisation for the time the process is running in the processor**. Please include these charts images for each algorithm in your pdf report.** In your report write 1:2 paragraphs for each algorithm, explaining the performance that you simulated. Your explanation should cover 1:2 examples of the process turnaround time that resulted in the average turnaround time for the algorithm and the total order of execution. Explain in 1:2 sentences how that would have changed if half of your processes blocked for I/O once during their execution giving an example of any order**. If you have 4 processes, come up with an example of 2 processes, say the first and the third, blocks for I/O half way through their execution, and put another estimate for the average turn-around time and final order of execution.**

PSS generates three text files in the output folder grouped in subfolders named after the algorithm abbreviated name. The first file is just an output of the configuration parameters. This file can help you verify that you configured your input file and initial values of the context switching and quantum correctly. The second file is a log of all state transitions occurred to every process. The third results file is where the performance metrics are calculated. It shows the timing every process started on the processor, and the time it finished, and hence calculates the turnaround time as the arrival time subtracted from the finish time. The weighted turnaround time for every process is its turnaround time divided by the expected run time (CPU Burst) that you defined in the input file as **y.** The average turnaround time is the total turnaround time for all processes divided by the number of processes. Same about average weighted turnaround time is the total for all processes divided by the number of processes. **Please include this output folder as a subfolder in your Android App source code for T2 zip file.**

T2: Android App Development - 75 marks

For the remaining learning objectives, you will need to build one Android app to do the following:

- Build a Mobile App that designates 2 folders in your internal memory as a source folder and a destination folder. There should be x files in the source folder, x should be the last digit in your URN, in the above example 8 files. If x in your ID is 0 or 1, you should put 2 files instead

in the source folder. Otherwise the number of the files should be exactly x. Your app should copy the x files from the source to the destination folders using 2 threads dividing the load (the number of files to copy from source to destination) between them. The 2 threads should not include the main UI thread.

- In the main UI thread, there is a shared variable about how many files have been already copied so far that is progressively updated in the UI to the user as the files are being copied. You will need to use proper synchronisation constructs to enable both threads to increment this variable and display it in the main activity UI as each thread progresses through its allocated files.

The mobile app will be evaluated from both your submitted source code, and the submitted coursework report document. Below is the details of what and where you should include and explain your work:

1. Thread Management and Synchronisation and concurrency – **25 marks**:
   a. **Source Code:** Correct Implementation of threads. **– 15 marks**
   b. **Report Document:** Time the scheduling and execution time of all threads, and explain that based on your understanding of Android Thread scheduling routines. Compare how this would have performed differently if same logic was implemented using the same Java concurrency models but in windows or in Ubuntu. – **5 marks**
   c. **Report Document:** Justify your choice of Java concurrency routines, and what other Java concurrency constructs are available and how they would have affected the performance. **– 5 marks**
2. Memory Management – **25 marks**:
   a. **Source Code:** Correct Memory allocation implementation – **15 marks**
   b. **Report Document:** Run your app while running a browser and a Facebook app simultaneously and keep alternating between each one of these 3 apps every 2 seconds. Explain how having three concurrent apps affected the memory available for each of them at the same time, and which one was swapped out first. Your explanation should consider garbage collection, context switching, memory management basics such as paging and segmentation, mapping logical to physical addresses by explaining your device physical memory size and virtual memory size and usage. – **10 marks**
3. I/O and File Management – **25 marks**:
   a. **Source Code:** Correct file management on different devices and user input implementation **– 15 marks**
   b. Report Document: Accessing files and folders and user input (I/O) and the required permissions. Explain how Android manages the files, buffering, and synchronisation, and file management in different types of devices. **- 10 marks**

Learning outcomes

	ILO	**Attributes Developed**
001	Understand the fundamental of Operating System principles, abstractions, mechanisms and their implementations.	KC
002	Design, develop and test a working application on a mobile device focusing on the use and primary functionalities of the Android OS.	KPT

003	Utilise the advanced features of Android OS in the context of how memory is managed, how tasks are scheduled, how interrupts are handled, file systems, I/O subsystems, etc. and put these concepts into practice by building (optimized) Android system applications.	KPT
004	Appreciate the complexities of OS issues (i.e., process and thread management, resource management and communication, etc.) in deploying commercial mobile applications and leading edge developments in the mobile application marketplace.	KC

Attributes Developed

C - Cognitive/analytical	T - Transferable skills
K - Subject knowledge	P - Professional/Practical skills

Generic learning outcomes for the COM1032 Mobile Computing Module

- Operating Systems Concepts
- Mobile Computing Concepts

Specific learning outcomes For OS part, out of 100 but weighted to 40% of the module, and assessed using a coursework

- Process Control and Scheduling 25%
- Threads & Concurrency 25%
- Memory Management 25%
- I/O and File Management 25%
- **Detailed Marks Distribution and Question Type in the OS coursework:**

	ILO	**Marks - 100**
1	**Process Control and Scheduling**	**25 marks**
101	Process States	4
102	Process Control Block	2
103	Process Privileges (Kernel vs User Space)	4
104	Process Scheduling	15
2	**Threads & Concurrency**	**25 marks**
201	Threads & Shared Memory	15
202	Threads States & Scheduling	5
203	Threads Synchronisation	5
3	**Memory Management**	**25 marks**
301	Memory Allocation & Relocation	15
302	Paging & Segmentation	5
303	Mapping Logical Addresses to Physical Addresses	5
304	Virtual Memory Employment	5
4	**I/O and File Management**	**25 marks**
401	I/O Function	15

402	Buffering	5
403	File Management	5
404	Sharing and Synchronisation	5

OS Marking Descriptors/Rubric (40% coursework):

ILO	A	B	C	D	E
1: Process Control and Scheduling – 25 marks					
101	Deep Understanding of the different Process States & the transitions	Fair Understanding of the process states & transitions	Missing Some essential concepts of the process states & transitions	Missed the most important definition of either the process states or the transitions or both	Poor or no understanding of the whole concept
102	Complete Understanding of the PCB with all its elements	Understands the most important components of the PCP	Missing some essential Components of PCB	Missed the most important component of the PCB	Poor or no understanding of the whole concept
103	Deep Understanding of kernel process & their privileges theoretically and the essential steps to create a new system call and interface it to user invocations	Complete Understanding of the kernel process & their privileges theoretically and the essential steps to create a new system call and interface it to user invocations	Missing some important concepts about kernel process & their privileges theoretically and the essential steps to create a new system call and interface it to user invocations	Missing the most important concepts about kernel process & their privileges theoretically and the essential steps to create a new system call and interface it to user invocations	Poor or no understanding of concepts about kernel process & their privileges theoretically and the essential steps to create a new system call and interface it to user invocations
104	Deep Understanding of process scheduling theoretically and the reports were accurate	Complete Understanding of the process scheduling, but abstract theoretical discussion	Missing some important process scheduling concepts, and poor theoretical	Missing the most important process scheduling algorithms, and very poor	Poor or no understanding of both the process scheduling algorithms and the theoretical

		and the reports are not 100% accurate	discussions – missing reports and accuracy	theoretical understandin g -	background – no reports or not accurate at all
2: Threads & Concurrency – 25 marks					
201	Clear Understandi ng of threads creation and management APIs – Program execute correctly	Some Understandi ng of the threads creation and managemen t APIs – Program execute with errors or could have been more efficient	Missing some important threads creation and manageme nt APIs – program does not execute but there is a lot of effort in the written code	Missing the most important threads creation and management APIs – program does not execute but there is little effort in the written code	Poor or no understandin g of the threads creation and management APIs – code is not interpretable and program does not execute
202	Clear Understandi ng of threads states and scheduling concepts	Some Understandi ng of the concept	Missing some important concepts	Missing the most important concept	Poor or no understandin g of the concept
203	Correct program that demonstrate s threads synchronisati on	Program does not compile, but genuine efforts have been put in the source code	Program does not compile, but code is missing some essential concepts	Program does not compile, but code is missing most important concepts	No Code submitted, or very poor attempt
3: Memory Management – 25 marks					
301	Correct program that demonstrate s Memory Allocation & Relocation– Program execute correctly	Program does not compile, but genuine efforts have been put in the source code – Program execute with errors or could have been more efficient	Program does not compile, but code is missing some essential concepts– program does not execute but there is a lot of effort in	– program does not execute but there is little effort in the written code	code is not interpretable and program does not execute

			the written code		
302	Clear Understanding of Paging & Segmentation concepts	Some Understanding of the concept	Missing some important concepts	Missing the most important concept	Poor or no understanding of the concept
303	Clear Understanding of Mapping Logical Addresses to Physical Addresses	Some Understanding of the concept	Missing some important concepts	Missing the most important concept	Poor or no understanding of the concept
304	Clear Understanding of Virtual Memory Employment theoretically and the essential commands	Some Understanding of the essential commands, but abstract theoretical discussion	Missing some important commands, and poor theoretical discussions	Missing the most important commands, and very poor theoretical understanding	Poor or no understanding of both the commands and the theoretical background
4: I/O and File Management – 25 marks					
401 & 403	Clear Understanding of I/O Function & File Management theoretically – Program executes correctly	Some Understanding of the concepts, but abstract theoretical discussion– Program execute with errors or could have been more efficient	Missing some important concepts, and poor theoretical discussions – program does not execute but there is a lot of effort in the written code	Missing the most important concepts – program does not execute but there is little effort in the written code	Poor or no understanding of both the concepts - code is not interpretable and program does not execute
402	Clear Understanding of I/O Buffering	Some Understanding of I/O Buffering	Missing Some essential concepts	Missed the most important concept	Poor or no understanding of the whole concept
404	Correct program that demonstrates File Sharing and	Program does not compile, but genuine efforts have	Program does not compile, but code is missing	Program does not compile, but code is missing most	No Code submitted, or very poor attempt

	Synchronisation	been put in the source code	some essential concepts	important concepts	

Report structure (Follow instructions carefully)

1. Include your details at the front and also include any references that you might have used at the end of the report.
2. The report should follow the task order and keep each task separated in a different section.
3. If you want to include more screenshots and results, please use the appendix.
4. Do not include the questions themselves in the report (only include the question numbers).
5. **Save the report as a pdf file before submission.**
6. **NOTE**: if you don't show screenshot(s) for a given task, you might be marked with 0 for that task.

If you fail to adhere to the report structure, you may receive a mark of 0 for each task affected.

Submitting the network files and report

You should submit your work to SurreyLearn in the Coursework submission folder. If you have any technical problems, you should try to submit via e-mail before the deadline. Please double check after submission to make sure your submission has indeed been uploaded to the system without any error.

Make sure to allow plenty of time before the deadline to submit your files. The network files should be zipped separately to the report pdf.

To prepare the network for submission to SurreyLearn:

1. Zip the entire coursework working folder including the source subfolders.
   ```
   $ zip –r username .
   ```
2. Submit the zip file and pdf report (separately) on SurreyLearn. The report needs to be submitted separately from the project zip file so that it can be checked for plagiarism automatically, or else it will NOT be accepted.

Your working folders provide evidence of your work. Without them, you will receive a mark of 0 for each task affected.

Late coursework policy

Note that late submissions will get 10% penalty per day and no submission will be allowed after the deadline + 3 days (72 hours) which means that you will get a zero mark if you submit more than 72 hours late. Read your programme handbook for further details about related regulations on coursework assessments.

Good Luck!

Manal Helal