UNIVERSITY OF JOHANNESBURG

FACULTY OF SCIENCE

ACADEMY OF COMPUTER SCIENCE AND SOFTWARE ENGINEERING	
MODULE	**IT08X37** **SYSTEMS PROGRAMMING**
CAMPUS	**APK**
EXAM SSA	**JANURARY 2019**

DATE	**2019-01-07**	**SESSION**	
ASSESSOR			**MR D COTTERRELL**
MODERATOR			**PROFESSOR BL TAIT**
		MARKS	**100**

INSTRUCTIONS

Please read the following instructions carefully:

1. Make sure that your project is burnt to CD and uploaded to EVE before the presentation starts.
2. Present all aspects of your practical project.
3. Clearly indicate the sources of all information used.
4. This paper consists of 6 pages (including cover page).

Student number:	
ID Number:	
Surname, Initials: (*Optional*)	

SM	
EM	
FM	

Professor B Tait

Mr. D Cotterrell

Notes/Comments

Criteria	Marks	

<u>Can be compiled and loaded as module</u>
• Appropriate Makefile **(1 marks)**
  o Produces .ko files for each module in solution
  o Uses the currently running kernels header files automatically. Marks should be deducted for hard coding the kernel version information.
  o Supports additional build targets such as clean
• Module parameters support **(1marks)**
• Device nodes are created in /dev/ **(2 marks)**
  o If created by script using mknod
  ▪ Appropriate file permissions should be set to allow for read/write access by not root users and groups.
  ▪ The device major/minor version numbers should be automatically parsed from /proc/devices and set without the need for user intervention.
  o If the device is implemented as a miscellaneous char device the device nodes will be automatically created. In this case only award **(1 mark)**.
  o If the student has researched an alternative in-kernel mechanism for creating the device nodes then full marks should be awarded for research effort **(2 marks)**.
• Device is correctly loaded into the kernel using insmod / can be queried modprobe
(1 marks)
	5	

<u>Can be compiled in kernel</u>
• Kernel build is successfully demonstrated **(3 marks)**
  o This does not require the kernel to be built in the presence of the marker, but a prebuilt kernel can be booted showing the presence / functioning of the device without the module being listed by lsmod.
• Process of compiling a module directly into the kernel is correctly explained. **(2 marks)**
  o These marks can be awarded even if the kernel build is not able to be successfully demonstrated.
User Mode:
• If the driver is user mode, the correct instruction must be shown to see how it has been added to the necessary registries and files in the Operating system. **(5 Marks)**
	5	

<u>Configuration</u> • Appropriate configuration options are presented to the user via make menuconfig or other similar kernel configuration tool. • The menu options should as a general rule correspond with the parameters accepted by the kernel module.	5	
<u>Uses appropriate Linux device model</u> • The student makes appropriate use of cdev family of functionality such as cdev_add() rather than register_chrdev*(). (If register _chrdev used (3 Marks)*	5	

Initialization • Function registered as init via module_init **(1 mark)** • Device(s) are correctly registered with the system **(1 mark)** • Error handling and reporting **(1 mark)** – *Although the kernel standard is to make use of a goto based style for error handling the Academy is perfectly willing to accept a more structured programming based approach which avoids gotos. As long as the code is clear and readable avoiding spaghetti code in the first case and overly deep nesting in the second no marks should be deducted.* • Major and or Minor numbers are dynamically assigned **(2 mark)**	5	
Shutdown • Function registered as exit via module_exit **(1 mark)** • Device(s) are correctly de-registered with the system **(1 mark)** • Error handling and reporting **(1 mark)** • Appropriate messages are sent via printk with appropriate severity levels **(2 marks)** – *This mark applies to messages sent during initialization and all other operations.*	5	
Open • Programmer defined data structure is correctly defined according to needs of system **(2 marks)**. • Memory is correctly allocated for data structure and associated with filep->private_data **(2 marks)** • Any per-reader/writer associated pre-initialization is performed **(1 mark).**	5	
Close • Memory is correctly de-allocated and any per-reader/writer cleanup performed. *Special attention should be paid to ensuring that internal dynamic data within the structure is deallocated prior to the structure to avoid any memory leaks.*	5	
Read/Write • File Operations structure instance is correctly declared and assigned the function addresses of the implementation operation functions. **(2 marks)** • Appropriate transfer of data to / from user and kernel space **(1 mark)** • Support for concurrency control to support multiple readers / writers **(2 marks)**	5	
User space dynamic interactively • Macros for each supported ioctl commands are correctly defined **(2 marks)**: *As a general rule there should at least be support for ioctl based read and write as well as triggering the major functionality of the module. Using ioctl to overshadow default or parameter based configuration options is also acceptable*. • The ioctl operation function is correctly implemented **(3 marks)**: *A rule the ioctl function should not actually implement the behaviors directly but should delegate to helper functions.*	5	

ELSE – if ioctl not used • Correct use of tools and approaches if ioctl is not applicable to driver developed, student must be able to justify approach taken. Also, student must show understanding of the purpose of the ioctl. **(5 Marks)**		
Virtual File System (/proc/ or sysfs) • Entries created in /proc/ or sysfs **(2 marks)** • Read / Write support of those entries depending on nature of entry **(2 marks)** • Explanation of difference between /proc/ and sysfs **(1 Mark)**	5	
User space program • Supports basic read / write **(2 marks)** • Supports ioctl based operations **(2 marks)** • Demonstrates functionality of module **(1 mark)**	5	
System utility • Meets requirements of problem domain	5	
System modularity • Source code modularity **(2 marks)** • Export of shared functionality via EXPORT_SYMBOL or another means to share functionality to different module **(3 marks)**	5	
Implementation of solution • Depending on the approach taken in developing the module: o The distribution of marks can be done based on the complexity of the implementation if it is purely a software based module. o If a hardware implementation the mark distribution will include evaluating whether the correct technology was used along with the corresponding libraries. **(20 marks)** • For example: o Hardware ▪ Correct operation of the device ▪ Correct choice of hardware parts and justification o Filesystem ▪ Unique file system features ▪ Mount and unmount o Ethernet ▪ Communication ▪ Broadcasting ▪ Multicasting o Network tty or ttysnooper ▪ Selection of tty device ▪ Can read directly from the terminal (pine) ▪ Can write directly to the terminal (commands, pine) ▪ (Network) ttys handled o Sound mixer ▪ Forward call system ▪ Selection of sound card ▪ Correctly listed subsystems	30	

▪ Super sampling , subsampling ▪ Conversion format ▪ Non-block / block IO ▪ Mmap • Does the implementation correctly handle multithreaded commands? **(5 marks)** • Quality of the implementation (doesn't reject data) **(5 marks)**		
Total	100	