Assignment One Theory Questions

Gearóid Sheehan R00151523

Operating Systems Engineering COMP8051

2 b) - What is a user program for the xv6 operating system? Explain how a user program links to library functions such as printf and how does it access the operating system?

A user program, or user process, in the xv6 operating system is a program which operates on top of the operating system. User programs are high level, sophisticated programs which allow the user to carry out complicated tasks through a well-designed user interface. They do not have direct access into the kernel however can communicate with it through the system call interface provided by the kernel. This is a set of predefined interfaces which allow user processes running in user-space to interact with the system using system calls such as read, write and exec. Library functions are written in pre-defined classes and are then declared in header files. User programs link to library functions such as printf by including the header files for the given library at the top of the class.

2 c) - Explain how the xv6 shell works – refer to the xv6 source code for your answer.

The shell in xv6 operating system is a user program for entering and calling commands. It typically handles tasks such as login sessions and runs other processes. It does so by taking user input and parsing those commands, before executing them. These commands are implemented using system calls convention such as read, write, fork, exec and wait. The code for the shell exists in sh.c. An input is read in using getcmd in a loop in the main method of sh.c, and the fork method then creates a copy of the shell process. The parent calls the wait method, and the child runs the command. So, if "ls" is entered into the shell as user input to the shell, runcmd is first called with "ls" as the argument. The exec method is then, and if it successfully runs the child process executes instructions from the ls.c class. When the exit system call is made, the child process ends, and the parent process returns from wait in main.

2 d) - Explain how xv6 implements the ls program – refer to the xv6 source code for your answer.

The ls command in xv6 is used to list files or directories in a current parent directory. When the command is entered, it prints all the said directories to the terminal where they can be further interacted with. In order to implement the ls command, xv6 includes a call to the ls user program, which is in the UPROGS section of the makefile so that it can be called from the terminal. The ls.c user program itself consists of a main method, the fmtname function and the ls function.

When ls is called from the terminal, the main method in ls.c takes the number of arguments given as well as a pointer to a char array of those arguments. If the arguments are less than 2 (the ls command itself is always the first argument) then the ls function is called with a full stop as the parameter (The ls function takes a char of a given directory as a parameter), as the current directory will be used in the command. If there are more arguments input alongside

the ls command, then a loop iterates over each of these arguments calling the ls function with the argument in each iteration.

The ls command, as mentioned above, takes a char of a directory as its parameter. Firstly, using an if statement it attempts to get a file descriptor for the given path. If this is unsuccessful, then an error is printed to the terminal and the function is exited. Otherwise, another if statement is used with the fstat function. Fstat is a function which gets the information about an open file associated with a given file descriptor (first parameter in the function call) and writes to the area pointed to by the second parameter of the function call. This area pointed to in the ls function is a struct called st of type stat. This struct is first declared in the stat.h file, which is included in the imports of ls.c. If the fstat function is successful then 0 is returned, and if this is not satisfied the said if statement prints an error to the screen, closes the file descriptor and exits the function.

Once these checks are passed, a switch statement exists in the ls function which checks the type variable in the st struct. If the type variable is of a file, then the fmtname function is used to get the name of the file, and the file name, type and file size. If the type variable is a directory, an if statement is used to check if the directory path given can fit into the declared buffer. If it fails, an error is printed to the terminal and the switch statement is exited. Otherwise, the path is copied to the buffer and a while loop uses the read function counts bytes from the file descriptor parameter to check if the count is equal to that of the struct de of type dirent. This struct is declared in the fs.h file. If the inum variable in de is equal to zero, the loop continues and uses the memmove function to copy the name string from the buffer to the de struct. The stat function is used in an if statement to list the properties of the given file path and struct. If the stat function fails and returns -1, then an error is printed. The fmtname function is used to print out the name of the file, and the file name, type and file size, same as done with a file type. The file descriptor is then closed, and the program exits in the main method.

4 a) - Read chapter 3 of the XV6 book and the xv6 source and describe how xv6 dispatches interrupts and system calls – refer to the xv6 source code for your answer.

In an operating system, an interrupt is a response by the processor to an event that need attention from the software. In xv6, the term interrupts refer to interrupts caused by devices not related to the current running process and are known as traps when referring to actual processes making the system call. Interrupt handlers are defined in the interrupt descriptor table (IDT) which has 256 entries. Devices on the motherboard can end up generating interrupts, which usually occur to inform the kernel that an I/O has completed. Interrupts are similar to system calls and share much of the code used for system calls and exceptions. They differ in that they can be generated at any given time and do not have to be called. Xv6 is designed for a board with multiple processes.

System calls are a set of predefined methods which user programs can call from the system call interface. These system calls allow the user programs to request operations from the kernel. In xv6, INT 64 is used for system calls, and an example of a system call is the write call. When printf is used to print a variable to the screen, the libc function is invoked. In the libc function, there is call to the write system call with the parameter STDOUT as a file descriptor. When this occurs, a software interrupt takes places it caused a transformation

from user space into kernel space and the operating system begins to execute. The operating then determines which system call has been made and invokes the handler for that system call. Once this has occurred, the correct peripherals are run (such as video card to display the given variable on the screen in the case of the write call with printf) and the control goes back into user space. The program will then continue to execute and finish.

4 b) - Explain how the keyboard driver buffers keystrokes for the xv6 operating system.

Universal Asynchronous Receiver Transmitter, or UART, is a physical circuit in a microcontroller with the main purpose of transmitting and receiving serial data. It is used by the xv6 operating system to take input from outside entities such as keystrokes from a keyboard. The UART is first initialized in the uartinit function where it sets up the bald rate. The interrupts are set up on COM1.

When keystrokes are typed in, they are written to a circular list buffer. The elements in the buffer are read using a FIFO system, and enter through the lists head and read out through its tail.