Linnéuniversitetet
Kalmar Växjö

Individual Assignment 1

Report and answers of all tasks of first assignment of Opration Systems 1DV512-HT21

Author: Saleh Shalabi

ss225bx@student.lnu.se

Semester: HT21

Course code: 1DV512

Date: 21/11/2021

1. Which programming languages were used to implement the Unix kernel? Why?

The C languge is used to Implement the most code of Unix kernel. But som of the code can not be writen in it, but Assembly. The kernel is about 10000 lines of code and 1000 of them is writen with assembly that could be only 800 lines that must be in assembly but for the sake of efficiency it still writen in it. Thise lines are not poosibole to write in C bucause the C lungage can not preform hardware functions.

2. Is the Unix kernel a complete operating system? How would you compare these concepts?

The Unix kernel is the heart “core” of the operating system. It’s not a complete operating system itself. That’s because that meny features that is founded in an operating system is messing from the kernel, such as that the Unix kernel doesn’t support file access methods or an operator’s console and meny more. Using the kernel as a tool meny of such features are implemented in the user software.

A complete operating system should offer the possibility to change the code to work with a ceratain task. The kernel can not be modified in anyway by the user, which means it’s to a specific way of work unlike an operating system.

3. If you decided to re-implement the Unix kernel using the modern programming languages and tools, which language(s) would you select and why?

To start with we can not write an kernel without using som Assembly code that is needed to access specific hard ware components that cant be accessed by any other language that cant be compailed to a machien language . The rest of the code could be almost writen in any lungage, but it deppends on why and what the kernel is writen for. Is it for an educating purpose or for fun then I think it can be writen in any modern language, this will make you learn more about the langauge limitations. In the other hand if it should be a proper kernel, then there is goals to reach such as bugs and security issuses even the efficiency of the operating system. I would use C or C++. C++ is an object oriented language that in my thoughts it may make it easier to implement the kernel, it might not be as fast as C but they

are almost the fastest langauges used today, or just use C to implement it because there is plenty of code of it that is open sourse that can be founded today and it will be to much help to se how some problems that may be encountered to during the work have been solved.

4. Are new running processes (programs) started from scratch in Unix? What is their relationship to the already running processes?

The new processes is created by the **fork()** operation, the origonl process is called “parent” and the new one is called “child” and it is a copy of the parent which means it inherits most of or all parent attributes or share no resourses. All processes have an identifier which is usualy an integert. All proceses have an parent process and can have meny child processes but process 0 wich would be created directly from the kernel[1]. The parent and the child most likely to share all of writable data, even opened files befor the fork are shared after it. The parent can be on wait until the termenation of it’s child, or continue to execute it’s programe concurrently. The child is created as a duplicate of the parent, which mean it would run the same program as its parent or it have a specific program loaded to run.

5. What are main classes of input-output devices supported by Unix? To which class would a Solid-State Drive (SSD) belong? How about a Brain-Computer Interface (BCI)?

The main classes of input-output devices supported in Unix are[2]:

- Block devices sush as flash memory or floppy disks, wich are usualy used to store file systems. The information in thies devices can be accessed in any order (randomly)
- Character devices sush as a keybord or mice. The main deffirens between the character and Block devices is that character devices doesn’t offer an randomly access but serially.

[1] Os Today (21/11/2021)
https://ostoday.org/other/your-question-what-is-parent-and-child-process-in-unix.html
[2] The operating system concepts. 9th Edition
A. Silberschatz, P. Galvin, G. Gagne

- Network devices.

The SSD is an Block device because it store data and offer randomly access to its information.

A BCI must be a character device that reades the brain signals to be transleted to comands, as in a keybord when pressing a button.

6. Does the Unix file system design use disks C: and D: (as in DOS or Windows)? Can a Unix system have several disk drives attached at the same time?

The Unix file system is organized into a hierarchy tree structure starting with the root directory. The directories in the Unix system are also files that user can't write. that means the Unix System does not use disks C: , D: and so on, each letter is considerd as a root in the Windows system, but in Unix all disks considerd as one logicale disk.

The Unix system can have several attached disks at the same time which will logically extends the Hierarchy bu munting the new disk drievs at any leaf.

7. What is an *i-node*? What are its contents and their purpose? Are i-nodes used for implementation of directories (if yes, how)?

The i-node is a 64-byte structure which defines a file in a list of file definitions called the i-list. Or it's a data structure for sorting file system metadata with pointers to its data[3].

It contains information about the file, including ownership, permissions, and location of the file contents. Each i-node contains the metadata needed to read a file. It contains 13 addresses on the disk, the 10 first points at the 10 blocks of a file and if its larger the next 3 will point at indirect blocks, and they are from eleventh to thirteenth single, duoble and triple indirect blocks. A single indirect block means an index block containing not data of the file but addresses of blocks that contains data.

[3] ibid.

Since directories are considered a type of files in Unix system so yes, they are indeed used in the implementation of it, that's because they as well will containe the file (in this case directorie's) metadata such ownership, size, permissions, location and even the addresses of the data in it.

8. What are the primitive file system operations supported in Unix?

The primitive file system operations are **create** which will need to have free space to allocate and add a new directory entry need to be made for the file even a pointer to the entry is placed in the system data segment for process **, open** which be a file handle provoider or a file descriptor to be used for other operations, even in the open a pointer will be set to the system data segment for the process**, write** wich will need to specify the name of the file and the information to be writen in the file and **read** the system call will need the name of the file and the place in the memory the file is stored to**, seek** wich will search the directory for an appropiate entry**, delete** to delete a file the directory is searched to find the file with the name specified, when it's founded the space of the file will be released så it could be reused by other files and the directory entry will be erased**,** and **truncate** which erases a content of file but not delete it, in this case the user saves the deletion and recreation of a file, and finaly **close** which basicly frees the structures built by create and open operations**.**