Homework Assignment #1 Solution

Q1. (Chapter 1, Problem 1) What are the two main functions of an operating system?

ANSWER: *An operating system performs two main functions: 1) providing application programmers a clean abstract set of resources instead of the messy hardware ones as an extended machine; and 2) managing hardware resources as a resource manager.*

Q2. What type of multiplexing (time or space) is mostly suitable for sharing the following resources, respectively: CPU, memory, disk, network card, printer, and keyboard?

ANSWER: *Time multiplexing: CPU, network card, printer, keyboard.*
Space multiplexing: memory, disk.

Q3. What are Program Counter (PC), Stack Pointer (SP), and Program Status Word (PSW) registers?

ANSWER: (*1) Program Counter register, or PC register, is a register containing the memory address of the next instruction to be fetched, i.e. a register containing the instruction address; (2) Stack Pointer register, or SP register, is a register containing the address of the top of the current stack in memory; (3) Program Status Word register, or PSW register, is a register containing the condition code bits, e.g. CPU priority, mode (user/kernel), which is needed in system calls and I/O.*

Q4. (Chapter 1, Problem 17) What is a trap instruction? Explain its use in operating systems.

ANSWER: *A trap instruction switches the execution mode of a CPU from the user mode to the kernel mode. This instruction allows a user program to invoke functions in the operating system kernel.*

Q5. What is a shell? Please name two shells.

ANSWER: *A shell is a command-line interface between the user and operating system, i.e. to let users to issue commands to interact with OS. Many shells exist and typical shells include bash, csh, and ksh (two examples would be sufficient).*

Q6. What is the purpose of a system call in an operating system?

ANSWER: *A system call allows a user process to access and execute operating system functions inside the kernel. User programs use system calls to invoke operating system services.*

Q7. (Chapter 1, Problem 23) A file whose file descriptor is *fd* contains the following sequence of bytes: 3, 1, 4, 1, 5, 9, 2, 6, 5, 3, 5. The following system calls are made:

lseek(fd, 3, SEEK_SET);
read(fd, &buffer, 4);

where the *lseek* call makes a seek to byte 3 of the file (note the file byte offset starts from 0). What does *buffer* contain after the read has completed?

ANSWER: *It contains the bytes: 1, 5, 9, 2.*

Q8. Based on Fig. 1-17, please briefly discuss these 11 steps involved in making the system call *read(fd, buffer, nbytes)*.

ANSWER: *Steps 1-3: Push parameters nbytes, &buffer, fd*
Step 4: Call the library procedure read
Step 5: The library procedure put a code for read in register
Step 6: Call trap instruction, trap to the kernel
Step 7: The kernel locates and dispatches the system call handler via a table of pointers to system call handlers indexed on system call number
Step 8: The system call handler runs
Step 9: Returns to the user-space library procedure
Step 10: The library procedure returns to the user program
Step 11: The SP is incremented (pop up, as stack grows downwards) to clean up the stack

Q9. (Chapter 1, Problem 26) In the example given in Fig. 1-17, the library procedure is called read and the system call itself is also called read. Is it essential that both of these have the same name? If not, which one is more important?

ANSWER: *It is more important for the library procedure to have that name. When the library procedure* `read` *traps to the kernel, it puts the code of the system call in a register. This code is used to index into a table. In other words, this code is critical to identify the system call being invoked, not the naming of the system call. On the other hand, the name of the library procedure is very important, since that is what appears in the program.*

Q10. Please list two disadvantages of a monolithic operating system structure and two advantages of a microkernel operating system structure.

ANSWER: *A monolithic OS structure can have issues/disadvantages like: 1) difficult to manage as it is essentially a collection of procedures linked together and visible to each other; and 2) vulnerable in terms of protection/security.*

A microkernel OS structure can have advantages like: 1) less buggy as the kernel is minimized; 2) less catastrophic even if an error/failure happens; and 3) high reliability and better security

protection.

THE END.