104062203 陳涵宇

Section 1. Problem Set

2.7: What are the two models of interprocess communication? What are the strengths and weaknesses of the two approaches?

The two models of interprocess communication are message-passing model and shared-memory model.

Message passing is useful for exchanging smaller amounts of data and easier to implement than is shared memory for intercomputer communication. But, message passing can handle only with small amounts of data.

Shared memory allows maximum speed and convenience of communication, but there are problems in the areas of protection and synchronization between the processes sharing memory.

2.10: What is the main advantage of the microkernel approach to system design? How do user programs and system services interact in a microkernel architecture? What are the disadvantages of using the microkernel approach?

The main advantage of the microkernel approach to system design is that it structures the operating system by removing all nonessential complements from the kernel and implementing them as system and user-level programs.

User programs and system services communicate to each other by message passing which is provided by microkernels.

Microkernels suffer from performance decrease due to increased system-function overhead.

Section 2.3. System Call Implementation

On line {test}/start.S:48, you have the MIPS assembly instruction
addiu $2, $0, SC_Halt
which is another way of saying
register2 = 0 + SC_Halt;
Explain the purpose of this assignment statement.

Before executing system call instruction, the register 2 will first be loaded with the integer that identifies which system call is invoked.

Section 2.4.1. Answer the following questions for the Write() system call.

1. How does the kernel get the pointer to the array of bytes to write? Give the description and C code.

When calling Write(), the pointer pointing to the message will be passed as an argument, instead of the message itself. Since the first argument will be saved at register 4, it is easy to get the pointer value there.

C code: val = kernel->machine->ReadRegister(4);

2. How does the kernel get the number of bytes to write? Give the description and C code.

The number of bytes to write is passed as second argument, thus we will get the value at register 5, according to the comment.

C code: bytes = kernel->machine->ReadRegister(4);

3. How does the kernel return the value to the caller? (for the number of bytes written?) Give the description and C code.

For any system call, the result, if any, must be put back into register 2.

C code: kernel->machine->WriteRegister(2, (int) bytes);