1. What is the difference between an operating system and middleware?

Middleware is computer [software](https://en.wikipedia.org/wiki/Software" \o "Software) that provides services to [software applications](https://en.wikipedia.org/wiki/Software_application" \o "Software application) beyond those available from operating system. It provide common services and capabilities to application other than what’s is offered by operating system.

operation system is a computer program that manages computer hardware and software resources.The operating system handles basic tasks such as managing and configuring memory, prioritizing the supply and demand of system resources, controlling input and output devices, operating networks, and managing file systems.The operating system also provides an interface for users to interact with the system.

1. What is the relationship between threads and processes?

Thread: Each process contains at least one thread, and these threads are sharing the resource space of the process, etc. When the thread changes, it will only cause changes to the process executed by the CPU, but will not change the resources owned by the process.Similarly, a program contains at least one process.The smallest unit of computation performed in a process and the basic unit of scheduling performed by the processor

Processes: Each process has its own address space. Resources such as memory, I/O,CPU, etc. Threads in the same process share the address space of the same process. Can you use the address space of other processes?Because of process independence, when a process crashes, it does not affect other processes in protected mode.The basic unit of resource allocation, the basic unit of operation scheduling, and the unit of concurrent execution in the system

1. Of all the topics previewed in chapter one of the text book, which one are you most looking forward to learning more about? Why?

I am most looking forward to learning more about Security. The security of the operating system is very, very important. The defects in operating system architecture.Operating system itself has the management of the management, CPU, memory, peripherals, every management involves some modules or procedures, if the problems in these applications, such as memory management problems, an external network connection to come over, just connect a defective modules, likely scenario is that the computer system will collapse.Therefore, some hackers are often aimed at the operating system is not perfect attack, so that the computer system, especially the server system immediately crashed

1. Suppose thread A goes through a loop 100 times, each time performing one disk I/O operation, taking 10 milliseconds, and then some computation, taking 1 millisecond. While each 10-millisecond disk operation is in progress, thread A cannot make any use of the processor. Thread B runs for 1 second, purely in the processor, with no I/O. One millisecond of processor time is spent each time the processor switches threads; other than this switching cost, there is no problem with the processor working on thread B during one of thread A's I/O operations. (The processor and disk drive do not contend for memory access bandwidth, for example.)
2. Suppose the processor and disk work purely on thread A until its completion, and then the processor switches to thread B and runs all of that thread. What will the total elapsed time be?

A: 100 \* ( 10 + 1 ) = 1100 ms

B: 1000ms

Total Time = 1100 + 1000 + 1 = 2101ms

The total elapsed time is 2101 milliseconds.

1. Suppose the processor starts out working on thread A, but every time thread A performs a disk operation, the processor switches to B during the operation and then back to A upon the disk operation's completion. What will the total elapsed time be?

Total Time = (1 + 9 + 1 + 1 ) \* 100 + 1 + (1000 - 9 \*100) = 1301ms

The total elapsed time is 1301 milliseconds.

1. In your opinion, which do you think is more efficient, and why?

The b has more efficient. When thread A take 10 millisecond for disk operation, thread A cannot make any use of the processor, so the processor can switch to thread B, to avoid wasting processor time.

1. Find and read the documentation for pthread\_cancel(). Then, using your C programming environment, use the information and the model provided in Figure 2.4 on page 26 of the text book to write a program in which the initial (main) thread creates a second thread. The main thread should sit on a read call of some kind, waiting to read input from the keyboard, waiting until the user presses the Enter key. At that point, it should kill off the second thread and print out a message reporting that it has done so. Meanwhile, the second thread should be in an infinite loop, each time around sleeping five seconds and then printing out a message. Try running your program. Can the sleeping thread print its periodic messages while the main thread is waiting for keyboard input? Can the main thread read input, kill the sleeping thread, and print a message while the sleeping thread is in the early part of one of its five-second sleeps?

The sleeping thread can print its periodic messages while the main thread is waiting for keyboard input.

The main thread can read input, kill the sleeping thread, and print a message while the sleeping thread is in the early part of one of its five-second sleeps

#include <pthread.h>

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

void \*thread\_function(void \*para)

{

    printf("the second thread start!\n");

    // each time around sleeping five seconds and then printing out a message.

    while (1)

    {

        sleep(5);

        printf("the second thread is running!\n");

    }

    printf("the second thread finish!\n");

    return 0;

}

int main()

{

    pthread\_t ntid;

    int err = pthread\_create(&ntid, NULL, &thread\_function, NULL);

    if (err != 0)

    {

        printf("can't create second thread\n");

        return -1;

    }

    printf("main thread: press any key to quit the second thread.\n");

    int c = getchar();

    pthread\_cancel(ntid);

    pthread\_join(ntid, NULL);

    return 0;

}

1. Suppose a system has three threads (T1, T2, and T3) that are all available to run at time 0 and need one, two, and three seconds of processing, respectively. Suppose each thread is run to completion before starting another. Draw six different Gantt charts, one for each possible order the threads can be run in. For each chart, compute the turnaround time of each thread; that is, the time elapsed from when it was ready (time 0) until it is complete. Also, compute the average turnaround time for each order. Which order has the shortest average turnaround time? What is the name of the scheduling policy that produces this order?

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 4 | 5 | 6 |
| T1 |  |  |  |  |  |  |
| T2 |  |  |  |  |  |  |
| T3 |  |  |  |  |  |  |

|  |  |
| --- | --- |
| thread | turnaround time |
| T1 | 1 |
| T2 | 3 |
| T3 | 6 |

The average turnaround time is 3.3333.

2)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 4 | 5 | 6 |
| T1 |  |  |  |  |  |  |
| T2 |  |  |  |  |  |  |
| T3 |  |  |  |  |  |  |

|  |  |
| --- | --- |
| thread | turnaround time |
| T1 | 1 |
| T2 | 4 |
| T3 | 6 |

The average turnaround time is 3.667.

3)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 4 | 5 | 6 |
| T1 |  |  |  |  |  |  |
| T2 |  |  |  |  |  |  |
| T3 |  |  |  |  |  |  |

|  |  |
| --- | --- |
| thread | turnaround time |
| T1 | 6 |
| T2 | 5 |
| T3 | 3 |

The average turnaround time is 4.667.

4)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 4 | 5 | 6 |
| T1 |  |  |  |  |  |  |
| T2 |  |  |  |  |  |  |
| T3 |  |  |  |  |  |  |

|  |  |
| --- | --- |
| thread | turnaround time |
| T1 | 6 |
| T2 | 5 |
| T3 | 3 |

The average turnaround time is 4.667.

1. Google the C standard library API and find out how to get information from the command line by using a printf() call to display a prompt, then another call [which you will look up] to get the user input. Write a program in C to prompt the user demographic information including name, age, class year, and any three other data times you wish. Structure the program as a call-and-response program such that each data item is a single question with a single answer. When all data has been obtained, display the data on the console. Each data item must be on a separate line, and it must be appropriately labeled. The output must be done using a single printf() statement.

#include <stdio.h>

#include <stdlib.h>

int main()

{

    char name[128];

    printf("Please input name:\n");

    scanf("%s", name);

    int age;

    printf("Please input age:\n");

    scanf("%d", &age);

    int class\_year;

    printf("Please input class year:\n");

    scanf("%d", &class\_year);

    int height;

    printf("Please input height:\n");

    scanf("%d", &height);

    int weight;

    printf("Please input weight:\n");

    scanf("%d", &weight);

    char city\_of\_birth[128];

    printf("Please input city of birth:\n");

    scanf("%s", city\_of\_birth);

    printf("current name is %s.\n", name);

    printf("current age is %d.\n", age);

    printf("current class year is %d.\n", class\_year);

    printf("current height is %d.\n", height);

    printf("current weight is %d.\n", weight);

    printf("current city of birth is %s.\n", city\_of\_birth);

    return 0;

}