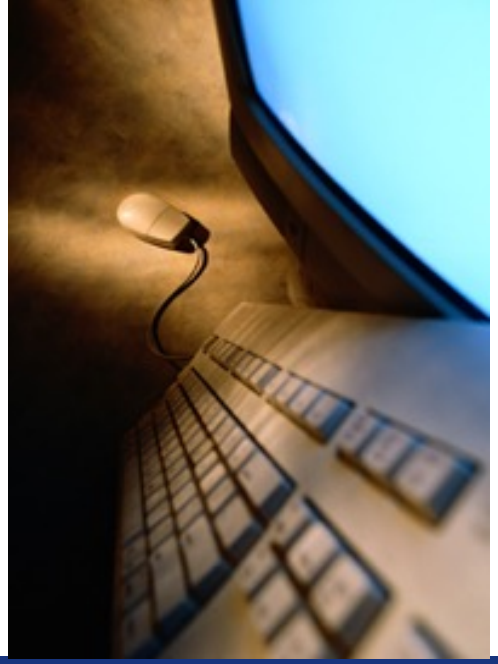
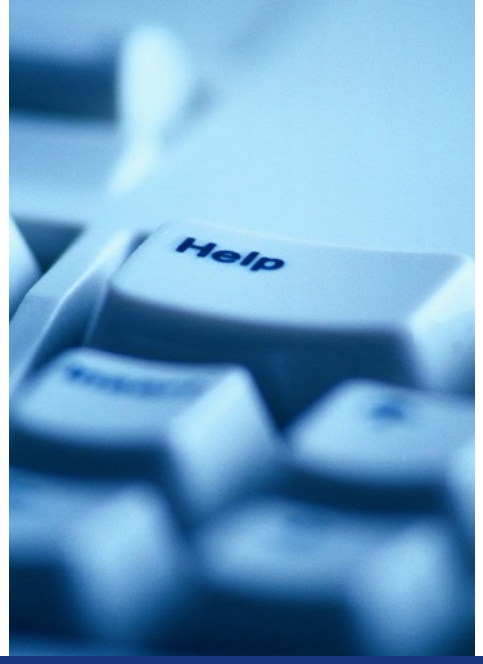




操作系统原理

Operating Systems Principles

陈鹏飞
计算机学院

作业-3

任务一：完成教材习题3.3、3.4、3.6

任务二：完成以下编程任务，并简要解析代码；

1.

Write a program that calls fork(). Before calling fork(), have the main process access a variable (e.g., x) and set its value to something (e.g., 100). What value is the variable in the child process? What happens to the variable when both the child and parent change the value of x?

fork() creates copy of parent process.
However, child and parent process have their own private address space exclusive of each other. Both process (child and parent) can't interfare in each other's address space (memory).

So, each maintain their own copy of variables.

作业-3

2.

Write a program that opens a file (with the open() system call) and then calls fork() to create a new process. Can both the child and parent access the file descriptor returned by open()? What happens when they are writing to the file concurrently, i.e., at the same time?

This Answer need verification Both child and parent can access the file descriptor opened using **open()**.

Both are able to write to the file but the order in which they do is un-deterministic (*if we don't use **wait()***).

Note: It may be possible that only one was able to write overwriting the other one.

作业-3

3.

Write a program that calls fork() and then calls some form of exec() to run the program /bin/ls. See if you can try all of the variants of exec(), including execl(), execle(), execlp(), execv(), execvp(), and execvP(). Why do you think there are so many variants of the same basic call?

Each variant serves its own purpose (altough they can be interchangbly used with little modification).

1. **execl** and **execv** are nearly identical (have very little difference).
2. **execlp** and **execvp** are nearly identical.
3. **execle** and **execvpe** are nearly identical.

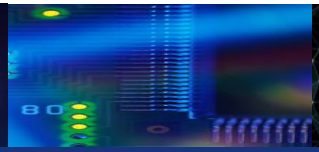
作业-3

4.

Now write a program that uses wait() to wait for the child process to finish in the parent. What does wait() return? What happens if you use wait() in the child?

wait(): on success, returns the process ID of the terminated child;
on error, -1 is returned.

If we use **wait()** in child process then **wait()** returns -1. Because, there is no child of child. So, there is no wait for any process (child process) to exit.



提交时间：4月2日

提交方式：http://course.dds-sysu.tech/course/3/homework