UC Day10

预习课

预习内容

信号基础

信号基础

信号基础

知识讲解

- 什么是信号
  - 信号在最早的UNIX系统中即被引入，已有30多年的历史，但只有很小的变化
  - 信号是提供异步事件处理机制的软件中断
  - 这些异步事件可能来自硬件设备，如用户同时按下了Ctrl键和C键，也可能来自系统内核，如试图访问尚未映射的虚拟内存，又或者来自用户进程，如尝试计算整数除以0的表达式
  - 进程之间可以相互发送信号，这使信号成为一种进程间通信(Inter-Process Communication, IPC)的基本手段

信号基础

- 什么是信号
  - 信号的异步特性不仅表现为它的产生是异步的，对它的处理同样也是异步的。程序的设计者不可能也不需要精确地预见什么时候触发什么信号，也同样无法预见该信号究竟在什么时候会被处理。一切都在内核的操控下，异步地运行。信号是在软件层面对中断机制的一种模拟

知识讲解

信号基础

知识讲解

- 信号的名称与编号
  - 信号是很短的消息，本质就是一个整数，用以区分代表不同事件的不同信号
  - 为了便于记忆，在signum.h头文件中用一组名字前缀为SIG的宏来标识信号，即为信号的名字
  - 通过kill -l 命令可以查看信号
  - 一共有62个信号，其中前31个信号为不可靠的非实时信号，后31个为可靠的实时信号

信号基础

- 常用的信号

编号	名称	说明	默认操作
1	SIGHUP	进程的控制终端关闭(用户登出)	终止
2	SIGINT	用户产生中断符(Ctrl+C)	终止
3	SIGQUIT	用户产生退出符(Ctrl+\)	终止+转储
4	SIGILL	进程试图执行非法指令	终止+转储
5	SIGTRAP	进入断点	终止+转储
6	SIGABRT	abort函数产生	终止+转储
7	SIGBUS	硬件或对齐错误	终止+转储
8	SIGFPE	算术异常	终止+转储
9	SIGKILL	不能被捕获或忽略的进程终止信号	终止
10	SIGUSR1	进程自定义的信号	终止
11	SIGSEGV	无效内存访问	终止+转储
12	SIGUSR2	进程自定义的信号	终止

信号基础

- 常用的信号

编号	名称	说明	默认操作
13	SIGPIPE	向读端已关闭的管道写入	终止
14	SIGALRM	alarm函数产生/真实定时器到期	终止
15	SIGTERM	可以被捕获或忽略的进程终止信号	终止
16	SIGSTKFLT	协处理器栈错误	终止
17	SIGCHLD	子进程终止	忽略
18	SIGCONT	进程由停止状态恢复运行	忽略
19	SIGSTOP	不能被捕获或忽略的进程停止信号	停止
20	SIGTSTP	用户产生停止符(Ctrl+Z)	停止
21	SIGTTIN	后台进程读控制终端	停止
22	SIGTTOU	后台进程写控制终端	停止
23	SIGURG	紧急I/O未处理	忽略
24	SIGXCPU	进程资源超限	终止+转储

信号基础

- 常用的信号

编号	名称	说明	默认操作
25	SIGXFSZ	文件资源超限	终止+转储
26	SIGVTALRM	虚拟定时器到期	终止
27	SIGPROF	实用定时器到期	终止
28	SIGWINCH	控制终端窗口大小改变	忽略
29	SIGIO	异步I/O事件	终止
30	SIGPWR	断电	终止
31	SIGSYS	进程试图执行无效系统调用	终止+转储

直播课见

UC

C/C++教学体系

目录

信号处理

信号处理

- 信号的处理
  - 忽略：什么也不做，SIGKILL(9)和SIGSTOP(19)不能被忽略
  - 默认：在没有人为设置的情况，系统缺省的处理行为
  - 捕获：接收到信号的进程会暂停执行，转而执行一段事先编写好的处理代码，执行完毕后再从暂停执行的地方继续运行

知识讲解

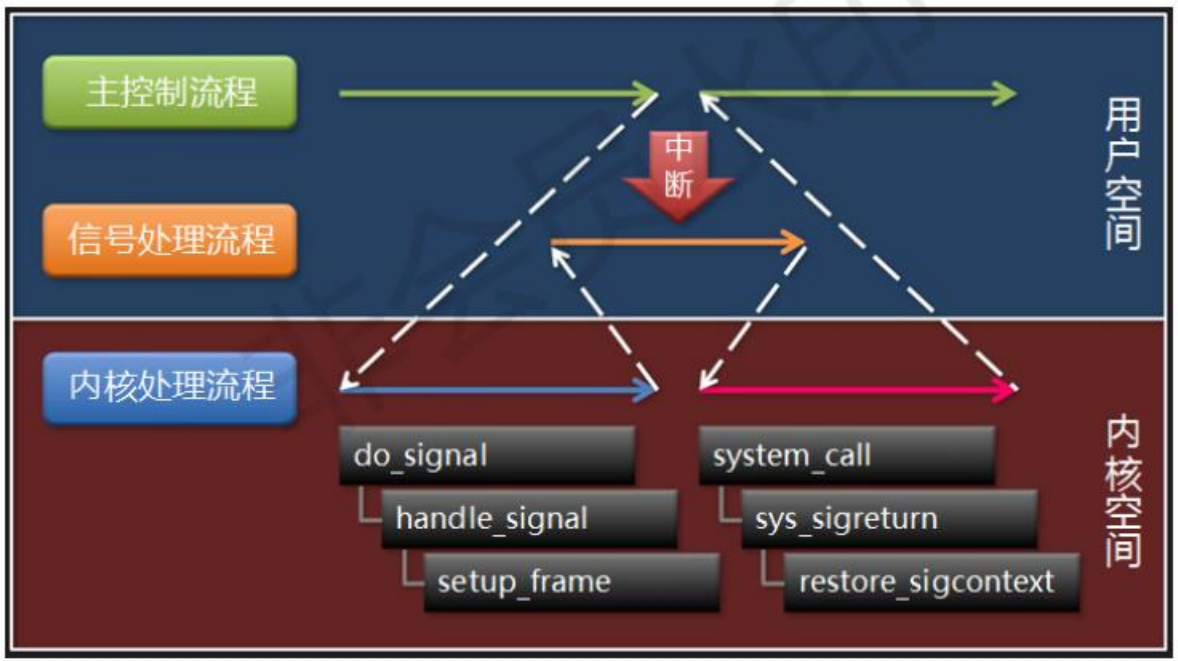
信号处理

知识讲解

- #include <signal.h>
- typedef void (*sighandler_t)(int);
- sighandler_t signal(int signum, sighandler_t handler);
  - 功能：设置调用进程针对特定信号的处理方式
  - 参数：signum 信号编号
    handler 信号的处理方式，可以如下取值
    SIG_IGN - 忽略
    SIG_DFL - 默认
    信号处理函数指针 - 捕获
  - 返回值：成功返回原信号处理方式，如果之前未处理过则返回NULL，失败返回SIG_ERR

信号处理

- 主控制流程、信号处理流程和内核处理流程



信号处理

- 主控制流程、信号处理流程和内核处理流程
  - 当有信号到来时，内核会保存当前进程的栈帧，然后再执行信号处理函数
  - 当信号处理函数结束后，内核会恢复之前保存的进程的栈帧，使之继续执行

知识讲解

太平间信号

太平间信号

- 如前所述，无论一个进程是正常终止还是异常终止，都会通过系统内核向其父进程发送SIGCHLD(17)信号。父进程完全可以在针对SIGCHLD(17)信号的信号处理函数中，异步地回收子进程的僵尸，简洁而又高效

```
void sigchld (int signum) {
        pid_t pid = wait (NULL);
        if (pid == -1) {
                perror ("wait");
                exit (EXIT_FAILURE); }
        printf ("%d子进程终止\n", pid);
    }
```

太平间信号

知识讲解

- 但这样处理存在一个潜在的风险，就是在sigchld信号处理函数执行过程中，又有多个子进程终止，由于SIGCHLD(17)信号不可靠，可能会丢失，形成漏网僵尸，因此有必要在一个循环过程中回收尽可能多的僵尸

```
for(;;){
    pid_t pid = waitpid(-1,NULL,WNOHANG);
    if(pid == -1){
        if(errno == ECHILD){
            printf("%d进程:没有子进程可回收\n",getpid());
            break;
        }else{
            perror("waitpid");
            return ;
        }
    }else if(pid == 0){
        printf("%d进程:子进程在运行,没法收\n",getpid());
        break;
    }else{
        printf("%d进程:回收路%d进程的僵尸\n",getpid(),pid);
    }
}
```

信号的发送

信号的发送

知识讲解

- 用专门的系统命令发送信号
  - kill [-信号] PID
  - 若不指明具体信号，缺省发送SIGTERM(15)信号
  - 若要指明具体信号，可以使用信号编号，也可以使用信号名称，而且信号名称中的“SIG”前缀可以省略不写。例如

    kill -9 1234

    kill -SIGKILL 1234 5678

    kill -KILL -1
  - 超级用户可以发给任何进程，而普通用户只能发给自己的进程

信号的发送

知识讲解

- #include <signal.h>
- int kill(pid_t pid, int signum);
  - 功能：向指定的进程发送信号
  - 参数：pid 可以如下取值

    -1 - 向系统中的所有进程发送信号

    >0 - 向特定进程(由pid标识)发送信号

    signum：信号编号，取0可用于检查pid进程是否存在，如不存在kill函数会返回-1，且errno为ESRCH
  - 返回值：成功(至少发出去一个信号)返回0，失败返回-1

信号的发送

知识讲解

- #include <signal.h>
- int raise (int signum);
  - 功能：向调用进程自己发送信号
  - 参数：signum 信号编号
  - 返回值：成功返回0，失败返回非0
  - kill(getpid(),signum) 等价于该函数

谢谢

UC Day010

复习课

信号处理的继承和恢复

信号处理的继承和恢复

- fork函数创建的子进程会继承父进程的信号处理方式。
  - 父进程中对某个信号进行捕获，则子进程中对该信号依然捕获
  - 父进程中对某个信号进行忽略，则子进程中对该信号依然忽略
- exec家族函数创建的新进程对信号的处理方式和原进程稍有不同
  - 原进程中被忽略的信号，在新进程中依然被忽略
  - 原进程中被捕获的信号，在新进程中被默认处理

知识讲解

下节课见