厦门大学《UNIX 系统程序设计》课程试卷

信息科学与技术学院 计算机 系 2013 年级 计算机科学与技术专业

学年学期：201520161 主考教师：王连生 A 卷（√）B 卷

一、 （36 分）简答下面问题：

（1） 如果两个常规文件的长度一样，那么是否意味着它们占用相同大小的磁盘空间？为什么？

（2） 读/写磁盘文件时，每次使用系统调用 read 和 write 是否都意味着读写磁盘驱动器？为什么？

（3） 如果有一个文件，它的权限被设置成文件的所有者不可读。那么文件的所有者是否能将其改为自己可读？如果其他用户可读，那么文件的所有者是否能将文件改为自己可读？

（4） 在作业控制中，进程组和进程对话有什么用途？

（5） 如果存在一个未决信号，如何在不执行相关的信号处理函数的前提下清除它？

（6） 为什么要避免进程僵尸？请给出两种避免进程僵尸的方法。

（7） 从 main 函数返回是否终止进程？函数 exit 与 _exit 有什么区别（至少指出两点）？

（8） 什么是系统调用？是否有在 C 语言标准里定义的库函数是系统调用？

（9） 如果删除一个已经打开的文件，那么能否继续读写这个打开的文件？

（10） 子进程的 ID 是大于 0 还是等于 0？它是否与父进程共享内存空间？它继承父进程对信号的处理方式吗？

（11） 什么是竞争条件（race condition）？使用信号机制是否可以避免？

（12） 函数不可重入的原因是什么？

二、 （15 分）阅读教材程序清单 4-7（p99），回答问题：

（1） dopath 函数返回值的有什么意义？

（2） 程序遍历目录树的次序是深度优先还是广度优先？为什么？

（3） 程序遍历目录树时，在每个节点执行的函数是什么？完成什么功能？

（4） 在 dopath 函数中，语句（不含引号）“ptr[-1] = 0;”是否可以省略？

（5） 在程序中是否可以使用系统调用 stat 获取目录项 i 节点的信息？为什么？

三 105

三、 （15 分）阅读下列程序代码，请说明第 13、14 行代码的目的是什么？

```
1   #include "apue.h"
2   #include <sys/wait.h>
3
4   int
5   main(void)
6   {
7       char    buf[MAXLINE];   /* from apue.h */
8       pid_t   pid;
9       int     status;
10
11      printf("%% ");  /* print prompt (printf requires %% to print %) */
12      while (fgets(buf, MAXLINE, stdin) != NULL) {
13          if (buf[strlen(buf) - 1] == '\n')
14              buf[strlen(buf) - 1] = 0;
15
16          if ((pid = fork()) < 0) {
17              err_sys("fork error");
18          } else if (pid == 0) {      /* child */
19              execlp(buf, buf, (char *)0);
20              err_ret("couldn't execute: %s", buf);
21              exit(127);
22          }
23
24          /* parent */
25          if ((pid = waitpid(pid, &status, 0)) < 0)
26              err_sys("waitpid error");
27          printf("%% ");
28      }
29      exit(0);
30  }
```

四、 （10 分）下面代码是利用 sigaction 函数实现 signal 函数的可靠信号版本。若使用这个 signal 函数，则

（1） 进入信号处理函数时，是否自动阻塞相同的信号？

（2） 进入信号处理函数后，是否自动恢复默认的信号处理方式？

（3） 若一个阻塞的低速系统调用被信号中断，那么这个系统调用是否重启？

（4） 若其子进程终止，则系统是否会向父进程发送信号（如果你认为会发送，则请说明发送什么信号）？

```
#include  <signal.h>
#include  "ourhdr.h"
sigfunc *signal(int signo, Sigfunc *func)
{
    struct sigaction act, oact;
    act.sa_handler = func;
    sigemptyset(&act.sa_mask);
    act.sa_flags = 0;
    if (signo == SIGALRM) {
#ifdef SA_INTERRUPT
        act.sa_flags |= SA_INTERRUPT;   /* SunOS */
#endif
    } else {
#ifdef SA_RESTART
        act.sa_flags |= SA_RESTART;     /* SVR4, 4.4BSD */
#endif
    }
    if (sigaction(signo, &act, &oact) < 0)
        return(SIG_ERR);
    return(oact.sa_handler);
```

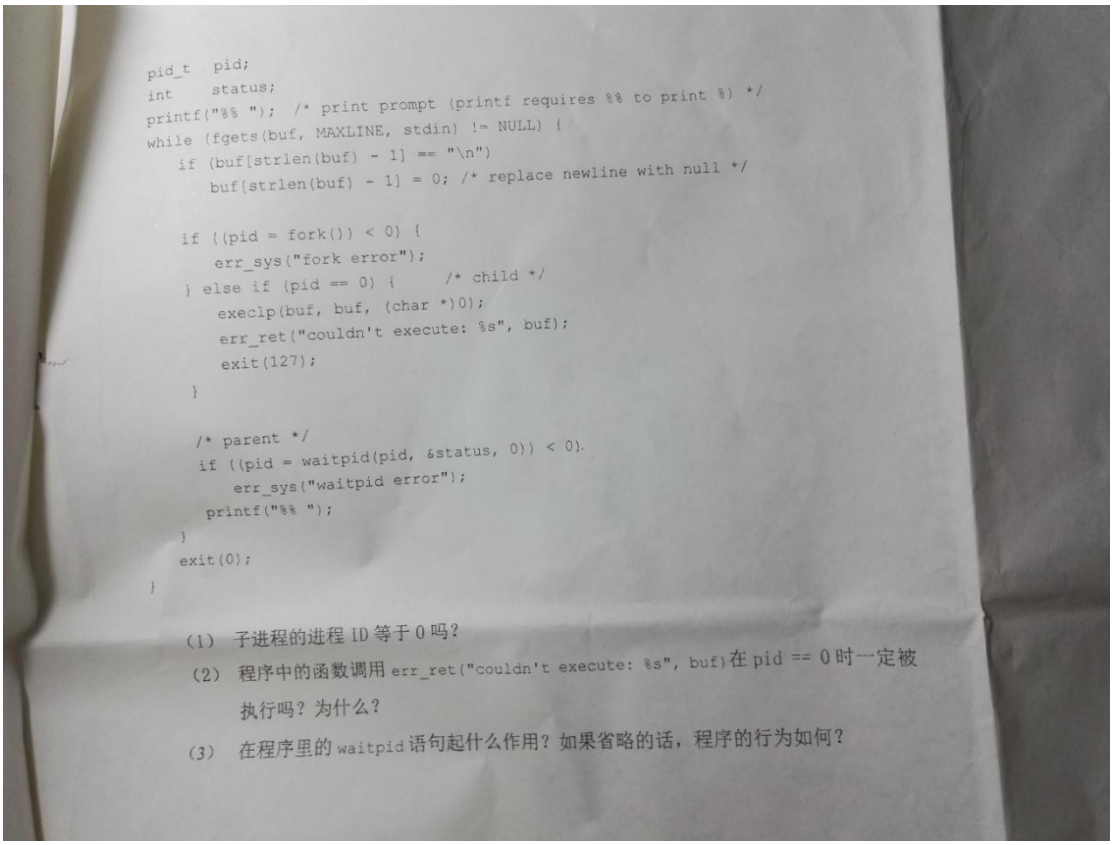
}

五、 （15 分）如果执行下面程序正常结束，则在标准输出上变量 glob 和 var 的输出值是什么？

```
#include "apue.h"
#include <sys/wait.h>

static int glob;

int
main(void)
{
    pid_t   pid;
    int     var;

    var = 0;
    if ((pid = fork()) < 0)
        err_sys("fork error");
    else if (pid == 0) {            /* child */
        glob++;
        exit(1);
    }
    if (wait(NULL) != pid)          /* wait for child */
        err_sys("wait error");

    if ((pid = vfork()) < 0)
        err_sys("fork error");
    else if (pid == 0) {            /* child */
        glob++;
        var++;
        exit(2);
    }
    printf("glob = %d   var = $d\n", glob, var);
    exit(0);
}
```

六、（9 分）阅读下面程序，简要回答问题：

```
#include "apue.h"
#include <sys/wait.h>

int main(void)
{
    char    buf[MAXLINE];   /* from apue.h */
```



```
pid_t   pid;
int     status;
printf("%% ");  /* print prompt (printf requires %% to print %) */
while (fgets(buf, MAXLINE, stdin) != NULL) {
    if (buf[strlen(buf) - 1] == "\n")
        buf[strlen(buf) - 1] = 0; /* replace newline with null */

    if ((pid = fork()) < 0) {
        err_sys("fork error");
    } else if (pid == 0) {      /* child */
        execlp(buf, buf, (char *)0);
        err_ret("couldn't execute: %s", buf);
        exit(127);
    }

    /* parent */
    if ((pid = waitpid(pid, &status, 0)) < 0)
        err_sys("waitpid error");
    printf("%% ");
}
exit(0);
}
```

(1) 子进程的进程 ID 等于 0 吗?

(2) 程序中的函数调用 err_ret("couldn't execute: %s", buf)在 pid == 0 时一定被执行吗? 为什么?

(3) 在程序里的 waitpid 语句起什么作用? 如果省略的话，程序的行为如何?

一、

（1） 不一定，因为 unix 允许文件中存在空洞，两个长度一样的文件若其中一个存在空洞，存在空洞的文件占用磁盘空间更小。

（2） 不一定，取决于用户缓冲区的大小和是否同步写，一般为了提高效率，系统通常是批量的读写磁盘数据存入缓冲区，以减少磁盘操作次数。

（3） 可以，可以。

（4） 对话是进程组的集合，它们都是作业控制系统的功能。进程可以同时向一族进程发信号，或者使用 waitpid 等待进程组中的一个子进程；一个对话将组成该对话的进程组与同一个控制终端联系起来。

我们可以将信号发送给一个进程组。进程组中的所有进程都会收到该信号。会话的意义在于将多个工作囊括在一个终端，并取其中的一个工作作为前台，来直接接收该终端的输入输出以及终端信号。其他工作在后台运行。

（5） 使用 sigwait 函数，将要清除的未决信号加入 sigwait 第一个参数 set 中。

将进程控制块中的信号处理方法单元对应的字段设置为 SIG_IGN。

（6） 危害是大量的僵尸进程会占用大量进程号，导致系统因为没有可用的进程号而不能产生新的进程。方法一是父进程通过 wait 和 waitpid 等函数等待子进程结束；方法二是用 signal(SIGCHLD,SIG_IGN)通知内核，这样子进程结束后，内核会回收。

（7） 不是，main 函数返回后会执行登记的 atexit 函数，进行释放内存之类的工作，之后结束程序并将 main 的返回值交给操作系统。区别一：_exit 函数直接调用 exit 系统调用关闭进程，而 exit 函数在调用 exit 系统调用前会先调用 atexit()注册的函数使用户可以在程序中之前执行自己的清理动作；区别二：exit 函数还会检查文件的打开情况，把文件缓冲区中的内容写回文件，清理 I/O 缓冲。

（8） 内核的接口被称为系统调用。C 语言标准里定义的函数都不是系统调用，但是可能该函数本身其实就是调用的系统调用。

（9） 能。

（10） 子进程 ID 大于 0。不共享内存空间。子进程会继承父进程对信号的处理方式。

（11） 两个或多个进程读写某些共享数据，而最后的结果取决于进程运行的精确时序，称为竞争条件。使用信号机制可以避免。

（12） 多个任务同时调用一个不可重入函数时，可能会改变其他任务调用这个函数时的数据，从而导致不可预料的后果。

a）已知它们使用静态数据结构；b）它们调用 malloc 或 free；c）它们是标准 I/O 函数。

二、

（1） 程序递归的遍历目录，返回值非 0 时结束递归遍历并且返回错误信息。

（2） 深度优先。

（3） 执行 myfunc 函数，根据第三个参数的不同，执行不同的操作。如果是 FTW_F 则将该节点对应的文件类型数量的变量加 1；如果是 FTW_D 则表示该文件是目录文件，将统计目录文件数量的变量加 1；如果是 FTW_DNR 显示错误信息“无法打开指定目录”；如果是 FTW_NS 显示错误信息“获取文件 stat 错误”。

（4） 不能省略。

（5） 不能，因为如果节点是个符号链接，stat 会返回该链接引用的文件的信息，而 lstat 会返回该符号链接的信息。

三、

fgets 会存下回车符，而我们模拟 shell 时，实际输入的命令是不包括回车符\n 的，第 13、14 行是判断 buf 最后一个字符是否为回车符，如果是的话将它去掉。

四、

（1） 是。

（2） 否。

（3） 如果是被 SIGALRM 中断则不重启，否则会重启。

（4） 会向父进程发送 SIGCHLD 信号（不确定，看不懂题目）。

五、

都是 1。

六、

（1） 不等于。

（2） 不一定，因为 execlp 函数执行成功的情况下是不会返回的，也就是执行成功的话下面的代码是不会执行的，只有失败了才会执行下面的 err_sys 函数。

（3） 等待进程 id 为 pid 的子进程结束，并将终止状态存放在 status 中；省略的话父进程可能会在子进程结束前进入下一轮 while 循环，可能产生僵死进程。