LINUX进程间通信：PIPE与FIFO

PIPE

http://ldl.wisplus.net/2010/10/01/linux%E8%BF%9B%E7%A8%8B%E9%97%B4%E9%80%9A%E4%BF%A1%EF%BC%9A%E7%AE%A1%E9%81%93/

概述：

int pipe(int pipefd[2]);
调用pipe函数在内核中开辟一块缓冲区(称为管道)用于单向通信,它有一个读端一个写端,然后通过filedes参数传给用户程序两个文件描述符,filedes[0]指向PIPE的读端,filedes[1]指向PIPE的写端。所以在用户程序看起来就像一个打开的文件,通过read(filedes[0]);
或者write(filedes[1]); 向这个文件读写数据其实是在读写内核缓冲区。

创建**PIPE**的基本步骤：

- 父进程调用pipe 开辟PIPE,得到两个文件描述符指向管道的两端。
- 父进程调用fork 创建子进程,那么子进程也有两个文件描述符指向同一管道。
- 父进程关闭管道读端,子进程关闭管道写端。父进程可以往PIPE里写,子进程可以从PIPE里读,PIPE是用环形队列实现的,数据从写端流入从读端流出,这样就实现了进程间通信

```c
#include<stdio.h>
#include<stdlib.h>
#include<unistd.h>
#include<sys/wait.h>
#include<string.h>

int main()
{
    char buf[20];
    pid_t pid;
    int fd[2];
    int n;

    pipe(fd); //创建管道
    if ((pid = fork()) < 0) //fork子进程
    {
        perror("fork error");
        exit(-1);
    } else if (pid == 0) //子进程中
    {
        close(fd[1]); //关闭写端
        n = read(fd[0],buf,20);
        write(STDOUT_FILENO,buf,n);
        close(fd[0]); //关闭读端
        exit(0);
    } else //父进程中
    {
        close(fd[0]); //关闭读端
        write(fd[1],"hello world",strlen("hello world"));
        close(fd[1]); //关闭写端
        waitpid(pid,NULL,0);
        exit(0);
    }
}
```

popen函数与**pclose**函数

标准IO函数库提供了popen函数,它创建一个管道并启动另外一个进程，该进程从该PIPE读出标准输入或将标准输出写入该PIPE。
FILE *popen(const char *command, const char *type);

int pclose(FILE *stream);
popen函数：先执行fork,然后调用exec(sh)以执行command,并且返回一个标准I/O文件指针。(错误返回NULL)
如果type是"r",则文件指针连接到command的标准输出,(该进程为读段,command所指进程为写端),参数"w"同理.
This command is passed to /bin/sh using the -c flag;
pclose函数：关闭由popen创建的标准I/O流,等待命令执行结束,然后返回**shell**的终止状态(错误返回-1)

```
#include<stdio.h>
#include<stdlib.h>
#include<unistd.h>

#define PAGER "${PAGER:-more}" //如果shell变量PAGER已经定义而且非空，则使用其值,否则使用字符串more
int main()
{
    char line[100];
    FILE *fpin,*fpout;
    if ((fpin = fopen("A.txt","r")) == NULL)
    {
       perror("can't open A.txt");
       exit(-1);
    }
    if ((fpout = popen(PAGER,"w")) == NULL)
    {
        perror("popen error");
        exit(-1);
    }
    while(fgets(line,100,fpin) != NULL)
    {
        if (fputs(line,fpout) == EOF)
        {
            perror("fputs error to pipe");
            exit(-1);
        }
    }
    if (pclose(fpout) == -1)
        perror("pclose error");
    exit(0);
}
```

FIFO

FIFO即是命名PIPE,文件系统中有个路径名与之关联。PIPE只能由有亲缘关系的进程使用,它们共同的祖先进程创建了管道。但是,通过FIFO,不相关的进程也能交换数据

创建**FIFO**

int mkfifo(const char *pathname, mode_t mode);
mode为存取许可权(需结合进程的umask).一般的文件I/O函数都可以用于FIFO
mkfifo函数已经隐含指定O_CREAT | O_EXCL,也就是说,要么创建一个新的FIFO，要么返回EEXIST错误(文件已经存在)

删除**FIFO**

int unlink(const char *pathname);
不同于PIPE，FIFO只有通过unlink才能从文件系统中删除

打开**FIFO**

int open(const char *pathname, int flags);
使用open函数打开FIFO,默认情况下没有指定O_NONBLOCK标志

```
//fifo_write.c
#include<stdio.h>
#include<stdlib.h>
#include<unistd.h>
#include<string.h>
#include<errno.h>
#include<fcntl.h>
#define FIFO_FILE "/tmp/myfifo"

int main()
{
    int fd = 0;
    int n;
    char buf[100];

```

我的随笔
我的评论
我的参与
最新评论
我的标签
更多链接

最新随笔
1. LINUX进程间通信：PIPE与FIFO
2. Android应用开发之（通过ClipboardManager, ClipData进行复制粘贴）
3. Linux下Socket编程
4. Good Topic, so forward ,tks to writer 无广告看视频
5. 数据结构使用教程 代码
6. 特殊字符在XML中的Unicode编码
7. Android开发日记 20110411-20110928
8. [转载]Android开发网上的一些重要知识点
9. Android 2.X Source Code 目录结构詳解
10. Android开源项目

随笔分类(19)
ASP(1)
ASP.Net
C#
C/C++(2)
IIS Apache Tomcat(1)
Java
Javascript
Linux(1)
PHP(1)
Protel 99SE
UML(4)
VB VB.net(1)
编程历程
单元测试
电路设计
敏捷开发(1)
嵌入式开发ARM(1)
软件工程(2)
设计模式(1)
收藏(3)
系统架构

随笔档案(260)
2012年8月 (3)
2012年7月 (1)
2011年11月 (1)
2011年10月 (1)

```
    if ((fd = open(FIFO_FILE,O_WRONLY | O_NONBLOCK)) < 0) //非阻塞方式打开
    {
        perror("open error");
        exit(-1);
    }
    while (1)
    {
        fgets(buf,100,stdin);
        n = strlen(buf);
        if ((n = write(fd,buf,n)) < 0)
        {
            if (errno == EAGAIN)
                printf("The FIFO has not been read yet.Please try later\n");
        }
    }
    return 0;
}

//fifo_read.c
#include<stdio.h>
#include<stdlib.h>
#include<unistd.h>
#include<sys/types.h>
#include<sys/stat.h>
#include<fcntl.h>
#include<errno.h>

#define FIFO_FILE "/tmp/myfifo"

int main()
{
    char buf[100];
    int n = 0;
    int fd;
    if ((mkfifo(FIFO_FILE,S_IRWXU) < 0) && (errno != EEXIST)) //如果该fifo文件不存在，创建之
    {
        perror("mkfifo error");
        exit(-1);
    }
    if ((fd = open(FIFO_FILE,O_RDONLY | O_NONBLOCK)) < 0) //非阻塞方式打开
    {
        perror("open error");
        exit(-1);
    }
    while (1)
    {
        if ((n = read(fd,buf,100)) < 0)
        {
            if (errno == EAGAIN)
            {
                printf("No data yet\n");
            }
        } else write(STDOUT_FILENO,buf,n);
        sleep(1); //sleep
    }
    unlink(FIFO_FILE);
    return 0;
}
```

FIFO与PIPE的读写：

(1)对于PIPE或FIFO的write总是往末尾添加数据，对他们的read则总是从开头返回数据。如果对PIPE或FIFO调用lseek,那就返回ESPIPE错误

(2)一个文件描述符能以2中方式设置成非阻塞：(默认为阻塞)

- 调用open是可指定O_NONBLOCK标志
- 如果文件描述符已经打开，那么可以调用fcntl设置O_NONBLOCK标志(PIPE只能采用这种方式)

(3)读写规则：

阻塞(缺省设置)：

只读open

- FIFO已经被只写打开：成功返回
- FIFO没有被只写打开：阻塞到FIFO被打开来写

2011年9月 (2)
2011年5月 (3)
2011年4月 (12)
2011年3月 (80)
2011年2月 (29)
2011年1月 (13)
2010年12月 (13)
2010年11月 (23)
2010年10月 (38)
2010年9月 (22)
2010年8月 (15)
2010年7月 (1)
2010年6月 (1)
2010年4月 (2)

文章档案(2)
2010年8月 (1)
2010年4月 (1)

相册(5)
pic(5)

友情链接
宁波企管
宁波企管
嵌入式Linux之我行
上海宏图条码

积分与排名
积分 - 45477
排名 - 5190

最新评论
1. Re:ajax系列教程，从读取，修改，添加，到酷酷的删除效果! - Web 开发 / Ajax
--Wehas Wang
2. Re:【收藏】FAT文件系统原理——MBR(主引导记录
好文章好文章
--放作夥
3. Re:android开发日记
good
--四海清一
4. Re:Android开源项目
good
--四海清一
5. Re:IT项目管理工具总结
好文章!
--风云

阅读排行榜
1. IT项目管理工具总结(18394)
2. Protel 介绍 protel99se正式汉化版下载 Protel DXP2004简体中文版(8005)

只写open
• FIFO已经被只读打开：成功返回
• FIFO没有被只读打开：阻塞到FIFO被打开来读
从空PIPE或空FIFO中read
• FIFO或PIPE已经被只写打开：阻塞到PIPE或FIFO中有数据或者不再为写打开着
• FIFO或PIPE没有被只写打开：返回0(文件结束符)
write
• FIFO或PIPE已经被只读打开：
写入数据量不大于PIPE_BUF(保证原子性):有足够空间存放则一次性全部写入,没有则进入睡眠，直到当缓冲区中有能够容纳要写入的全部字节数时，才开始进行一次性写操作
写入数据量大于PIPE_BUF(不保证原子性):缓冲区一有空闲区域，进程就会试图写入数据，函数在写完全部数据后返回
• FIFO或PIPE没有被只读打开：给线程产生SIGPIPE(默认终止进程)

O_NONBLOCK设置：
只读open
• FIFO已经被只写打开：成功返回
• FIFO没有被只写打开：成功返回
只写open
• FIFO已经被只读打开：成功返回
• FIFO没有被只读打开：返回ENXIO错误
从空PIPE或空FIFO中read
• FIFO或PIPE已经被只写打开：返回EAGAIN错误
• FIFO或PIPE没有被只写打开：返回0(文件结束符)
write
• FIFO或PIPE已经被只读打开：
写入数据量不大于PIPE_BUF(保证原子性):有足够空间存放则一次性全部写入,没有则返回EAGAIN错误(不会部分写入)
写入数据量大于PIPE_BUF(不保证原子性):有足够空间存放则全部写入,没有则部分写入,函数立即返回
• FIFO或PIPE没有被只读打开：给线程产生SIGPIPE(默认终止进程)

PIPE或**FIFO**若干额外的规则：
• 如果请求读取的数据量多余当前可用的数据量，那么返回这些可用的数据
• 如果请求写入的数据字节数小于或等于PIPE_BUF,那么write操作保证是原子的(**O_NONBLOCK**标志的设置对原子性没有影响)
• 当对PIPE或FIFO最后一个关闭时，仍在该PIPE或FIFO上的数据将被丢弃

FIFO与**PIPE**的限制：

• 它们是半双工的(单向性),即数据只能在一个方向上流动。由进程A流向进程B或由进程B流向进程A。
• PIPE的读写端通过打开的文件描述符来传递,因此要通信的两个进程必须从它们的公共祖先那里继承PIPE文件描述符。FIFO可以实现无关进程间的通信。
• 一个进程在任意时刻打开的最大文件描述符个数OPEN_MAX(通过调用sysconf(_SC_OPEN_MAX)获得)
• 可原子地写往PIPE或FIFO的最大数据量PIPE_BUF(通常定义在limits.h)

小结：

• PIPE普遍用于SHELL中，不过也可以从程序中使用，往往是从子程序向父程序回传信息。使用PIPE时涉及的某些代码(pipe、fork、close、exec和waitpid)可通过使用popen和pclose来避免，由它们处理具体细节并激活一个shell
• FIFO与管道类似，但他们是用mkfifo创建的，之后需要用open打开。打开管道时必须小心，因为有许多规则制约着open的阻塞与否(甚至发生死锁)

山 人
关注 - 46
粉丝 - 33
+加关注

« 上一篇：Android应用开发之（通过ClipboardManager, ClipData进行复制粘贴）

posted @ 2012-08-05 12:50 山 人 阅读(4119) 评论(0) 编辑 收藏

3. Lua 5.1 参考手册(4849)
4. Keil uVision3下载 (破解版带注册机+中文版)(4259)
5. LINUX进程间通信：PIPE与FIFO(4119)

评论排行榜
1. IT项目管理工具总结(27)
2. Eclipse For PHP Debug 设置记录 (备忘)(4)
3. IceScrum2 for Agile Development Introduction(4)
4. 2010年最怪异的25个面试问题，你能回答吗？(3)
5. 深入理解ASP中FSO的神奇功能(3)

推荐排行榜
1. IT项目管理工具总结(11)
2. IceScrum2 for Agile Development Introduction(3)
3. (收藏）JQuery使用手册(3)
4. Eclipse For PHP Debug 设置记录 (备忘)(3)
5. 深入理解ASP中FSO的神奇功能(2)