实验 7 嵌入式 LINUX 文件 I/O 实验

实验目的

- 掌握嵌入式 LINUX 文件 I/O 编程方法
- 掌握嵌入式 LINUX 应用程序编译、运行方法

实验设备

- 硬件：PC 机+Tiny6410 开发板
- 软件：WINXP+VMWARE 虚拟机+UBUNTU10.04-32bit+arm-linux-gcc

实验内容

- 1.编写程序 embed_file1.c，embed_file2.c，embed_file3.c，embed_file4.c，实现文件的打开、读写、关闭操作；其中这 4 个程序分别完成如下功能：

embed_file1.c.创建一个名字为“hello”的文件，并写入 26 个大写的英文字母，关闭该文件；
embed_file2.c.打开“hello”文件，从文件开头第 5 个字符位置开始，读出 10 个字符出来；
embed_file3.c.打开“hello”文件，从文件开头第 20 个字符位置开始，读出 10 个字符出来；
embed_file4.c.把“hello”文件拷贝成“hello.backup”文件

- 2.编写 Makefile 文件，交叉编译该程序，把生成的 4 个可执行程序下载到开发板执行，观看运行结果。

实验原理

1. Open 函数

```
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
int open(const char* pathname, int oflag, …/*mode_t mode */)
```

2.close 函数

#include <unistd.h>
int close (int *filedes*)

3.lseek 函数

```
#include <sys/types.h>
#include <unistd.h>
off_t lseek(int filesdes, off_t offset, int whence) ;
```

4.read 函数

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

ssize_t read(int *feledes*, void* *buff*, size_t *nbytes*) ;

5.write 函数

#include <unistd.h>

ssize_t write(int *filedes*, const void * *buff*, size_t *nbytes*) ;

实验步骤

1) 在虚拟 LINUX 下编辑源程序 embed_file1.c，embed_file2.c，embed_file3.c，embed_file4.c

2) 在虚拟 LINUX 下编辑 Makefile 文件，注意：这里编译器使用交叉编译工具 arm-linux-gcc，

3) 在虚拟 LINUX 下执行 make，把 embed_file1.c，embed_file2.c，embed_file3.c，embed_file4.c 交叉编译成 4 个可执行文件 embed_file1，embed_file2，embed_file3，embed_file4。（该步骤也可以使用交叉编译工具 arm-linux-gcc 直接把源文件编译成可执行程序）

4) 把 4 个可执行文件 embed_file1，embed_file2，embed_file3，embed_file4 从虚拟 LINUX 拷贝到 windows 下，在虚拟 LINUX 这四个文件所在目录执行如下命令：

cp embed_file1 embed_file2 embed_file3 embed_file4 /mnt/hgfs/share

指令执行成功后，会在 WINDOWS 的“D:\VM_OS\ub100432bit\share”目录下看到这四个文件。

5) 打开 WINDOWS 的超级终端（超级终端配置参考“Tiny6410 设置超级终端.pdf”）。使用串口线把开发板的 COM0 口和 PC 的串口连接起来，把开发板的 S2 开关拨到 NAND 一侧，连接电源线，然后上电启动开发板，此时开发板上的 LINUX 系统开始启动，会在超级终端看到启动信息。

6) 等到开发板的 LINUX 系统启动完成后，使用超级终端，通过串口把 4 个可执行文件下载到开发板：超级终端传送菜单-→发送文件，打开发送文件对话框，点击文件名后的浏览按钮，找到要发送的文件，协议选择 Zmodem 与崩溃恢复，点击发送，文件就发送到嵌入式 LINIUX 的当前目录下。注意：4 个可执行文件要依次发送。注意：如果想重新发送某文件，应先把开发板上原来的文件删掉，然后再发送。新发送的文件不会覆盖开发板上原来的同名文件。

以下步骤所涉及到的操作，均针对开发板上的嵌入式 LINUX 操作系统，因此以下步骤涉及到的命令均在超级终端窗口下达。

7) 在超级终端中下达命令（此时的超级终端就是开发板上的 LINUX 的默认输出设备，可通过超级终端对开发板上的 LINUX 系统下达指令），对四个可执行文件增加可执行权限“chmod +x embed_file1 embed_file2 embed_file3 embed_file4”，然后依次执行 embed_file1，embed_file2，embed_file3，embed_file4。

./embed_file1，./embed_file2，./embed_file3，./embed_file4

8) 数据记录：每执行一个可执行文件，记录输出信息

参考源程序

embed_file1.c 源程序：

```
#include <unistd.h>
#include <sys/types.h>
#include <sys/stat.h>
```

```
#include <fcntl.h>
#include <stdlib.h>
#include <stdio.h>
#include <string.h>

void main()
{
    char w_buf[]="ABCDEFGHIJKLMNOPQRSTUVWXYZ";
    int filedes;

    if((filedes=open("hello",O_RDWR|O_CREAT,0644))==-1)
    {
        perror("open hello WRONG!");
        exit(1);
    }
    if(write(filedes,w_buf,26)==-1)
    {
        perror("write hello WRONG!");
        exit(1);
    }
}
```

embed_file2.c 源程序

```
#include <unistd.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <stdlib.h>
#include <stdio.h>
#include <string.h>

#define READ_SIZE 10
void main()
{
    char r_buf[READ_SIZE];
    int filedes;

    if((filedes=open("hello",O_RDWR))==-1)
    {
        perror("open hello WRONG!");
        exit(1);
    }
    if(lseek(filedes,5,SEEK_SET)==-1)
    {
```

```
            perror("lseek hello WRONG!");
            exit(1);
        }
        if(read(filedes,r_buf,10)==-1)
        {
            perror("read hello WRONG!");
            exit(1);
        }
        else
            printf("%s\n",r_buf);
}
```

embed_file3.c 源程序

```
#include <unistd.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <stdlib.h>
#include <stdio.h>
#include <string.h>

#define READ_SIZE 10
void main()
{
        char r_buf[READ_SIZE];
        int filedes;

        if((filedes=open("hello",O_RDWR))==-1)
        {
            perror("open hello WRONG!");
            exit(1);

        }
        if(lseek(filedes,20,SEEK_SET)==-1)
        {
            perror("lseek WRONG!");
            exit(1);
        }
        if(read(filedes,r_buf,10)==-1)
        {
            perror("read hello WRONG!");
            exit(1);
        }
        else
```

```
        printf("%s\n",r_buf);
}
```

embed_file4.c 源程序

```
#include <unistd.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <stdlib.h>
#include <stdio.h>
#include <string.h>

void main()
{
        char buf[1];
        int fd1,fd2;

        if((fd1=open("hello",O_RDWR))==-1)
        {
                perror("open hello WRONG!");
                exit(1);

        }

        if((fd2=open("hello.backup",O_RDWR|O_CREAT,0644))==-1)
        {
                perror("open hello.backup WRONG!");
                exit(1);

        }

        if(lseek(fd1,0,SEEK_SET)==-1)
        {
                perror("lseek hello WRONG!");
                exit(1);
        }

        while(read(fd1,buf,1)!=0)
        {
                if(write(fd2,buf,1)==-1)
                {
                        perror("write hello.backup WRONG!");
                        exit(1);
                }
```

```
    }
}
```

Makefile

```
.PHONY:all clean
OBJS    =   embed_file1.o embed_file2.o embed_file3.o embed_file4.o
CC      =   arm-linux-gcc
TARGETS     =embed_file1 embed_file2 embed_file3 embed_file4

all:$(TARGETS)
$(TARGETS):%:%.o
    $(CC) -o $@ $<
$(OBJS):%.o:%.c
    $(CC) -c -o $@ $<

clean:
    rm $(TARGETS) $(OBJS)
```