Linux File I/O : Reading and Writing

Advanced Embedded Linux Development

with **Dan Walkes**

University of Colorado **Boulder**

Learning objectives:

Reading and Writing Files

How umask influences file permissions

Introduction to strace

Reading file content

- read()
- Returns:
  - Number of bytes read, or -1 on error (and sets errno)
  - Return values < count mean no more bytes are available now (end of file, interrupted by signal)

```
READ(2)                                                  Linux Programmer's Manual

NAME
       read - read from a file descriptor

SYNOPSIS
       #include <unistd.h>

       ssize_t read(int fd, void *buf, size_t count);

DESCRIPTION
       read() attempts to read up to count bytes from file descriptor fd into the buffer starting at buf.
```

http://man7.org/linux/man-pages/man2/read.2.html

Read File Simple Example

```c
int main(int argc, char **argv)
{
    int rc=-1;
    if( argc != 2 ) {
        printf("Usage: %s filename\n",argv[0]);
        printf("    The file specified as \"filename\" will be created with zero size if it does not exist.\n");
    } else {
        const char *filename = argv[1];
        int fd=open(filename,
                O_RDWR|O_CREAT,
                S_IRWXU|S_IRWXG|S_IRWXO);
        char buff[1];
        if( fd == -1 ) {
            printf("Error opening %s\n",filename);
        } else {
            int readlen = read(fd,buff,1);
            rc = 0;
            if( readlen == -1 ) {
                printf("Error with read(), errno is %d (%s)\n",errno,strerror(errno));
                rc = -1;
            }
            if( readlen != 0 ) {
                printf("read %d bytes, first byte was %c\n",readlen,buff[0]);
            }
            else {
                printf("Read 0 bytes (empty file)\n");
            }
            close(fd);
        }
    }
    return rc;
}
```

- Read the first byte of a file
- Create the file with zero size if it doesn't exist.

https://github.com/cu-ecen-5013/aesd-lectures/blob/b85f25650e4de21985145ba88ccfa1a9bbe276f7/lecture5/readfile.c

Create File and Umask

```
aesd@aesd-VirtualBox:~/aesd-lectures/lecture5$ ./run-empty-file-read.sh
Read 0 bytes (empty file)
aesd@aesd-VirtualBox:~/aesd-lectures/lecture5$ ls -l
total 24
-rwxrwxr-x 1 aesd aesd    0 Jan 26 18:06 emptyfile
-rwxrwxr-x 1 aesd aesd 8624 Jan 26 18:17 readfile
-rw-rw-r-- 1 aesd aesd 1139 Jan 26 18:14 readfile.c
-rwxrwxr-x 1 aesd aesd   59 Jan 26 18:14 run-empty-file-read.sh
-rwxrwxr-x 1 aesd aesd  181 Jan 26 18:14 run-pipe-read.sh
```

- RWX for others?

```
int fd=open(filename,
        O_RDWR|O_CREAT,
        S_IRWXU|S_IRWXG|S_IRWXO);
```

- What happened to the write permission?
  - Cleared by umask
- What is umask?
  - Per-process way to restrict permissions set on created files. (& ~umask)

```
aesd@aesd-VirtualBox:~/aesd-lectures/lecture5$ umask
0002
```

https://github.com/cu-ecen-5013/aesd-lectures/blob/master/lecture5/readfile.c

Read File Simple Example

- What happens if I change my umask to 0003?

```
aesd@aesd-VirtualBox:~/aesd-lectures/lecture5$ umask 0003
aesd@aesd-VirtualBox:~/aesd-lectures/lecture5$ umask
0003
aesd@aesd-VirtualBox:~/aesd-lectures/lecture5$ rm emptyfile
aesd@aesd-VirtualBox:~/aesd-lectures/lecture5$ ./run-empty-file-read.sh
Read 0 bytes (empty file)
```

```
aesd@aesd-VirtualBox:~/aesd-lectures/lecture5$ ls -l
total 24
-rwxrwxr-- 1 aesd aesd    0 Jan 26 18:22 emptyfile
-rwxrwxr-- 1 aesd aesd 8624 Jan 26 18:22 readfile
-rw-rw-r-- 1 aesd aesd 1139 Jan 26 18:14 readfile.c
-rwxrwxr-x 1 aesd aesd   59 Jan 26 18:14 run-empty-file-read.sh
-rwxrwxr-x 1 aesd aesd  181 Jan 26 18:14 run-pipe-read.sh
```

```
int fd=open(filename,
        O_RDWR|O_CREAT,
        S_IRWXU|S_IRWXG|S_IRWXO);
```

- Now both w and x are cleared.

https://github.com/cu-ecen-5013/aesd-lectures/blob/master/lecture5/readfile.c

System Call Logging with strace

```
aesd@aesd-VirtualBox:~/aesd-lectures/lecture5$ strace ./readfile emptyfile
execve("./readfile", ["./readfile", "emptyfile"], 0x7ffe16ee9088 /* 22 vars */) = 0
brk(NULL)                               = 0x563b0f645000

write(1, "Opening emptyfile in blocking mo"..., 35Opening emptyfile in blocking mode
) = 35
openat(AT_FDCWD, "emptyfile", O_RDWR|O_CREAT, 0777) = 3
write(1, "Reading 1 byte from emptyfile\n", 30Reading 1 byte from emptyfile
) = 30
read(3, "", 1)                          = 0
write(1, "Read 0 bytes (empty file)\n", 26Read 0 bytes (empty file)
) = 26
close(3)                                = 0
```

- strace is a handy tool for debugging application interaction with the kernel through syscall
- No source code required!

https://github.com/cu-ecen-5013/aesd-lectures/blob/master/lecture5/readfile.c

Non-Blocking Reads

```
int fd=open(filename,
        O_RDONLY|
        /**
        * When opening in blocking mode, use O_CREAT|O_RDWR to support creating the file if it doesn't exist
        * and to allow opening pipes on Linux when no readers are available (see http://man7.org/linux/man-pages/man7/fifo.7.html)
        * blocking mode.
        */
        block ? O_CREAT|O_RDWR : O_NONBLOCK,
        S_IRWXU|S_IRWXG|S_IRWXO);
```

Use O_NONBLOCK on open to open in non blocking mode

```
int readlen = read(fd,buff,1);
rc = 0;
if( readlen == -1 ) {
    if( !block && errno == EAGAIN ) {
        printf("EAGAIN returned from read(), no data available to read\n");
    } else {
        printf("Error returned from read(), errno is %d (%s)\n",errno,strerror(errno));
    }
```

check for EAGAIN to see if data is available

- O_NONBLOCK allows your thread needs to do work while waiting for fifo/pipe data.

Linux System Programming Chapter 2
https://github.com/cu-ecen-5013/aesd-lectures/blob/master/lecture5/readfile.c

Non-Blocking Reads Example

```
aesd@aesd-VirtualBox:~/aesd-lectures/lecture5$ ./run-pipe-read.sh
make: Nothing to be done for 'all'.
Attempting read on a pipe, will block
Run "echo 'hi' > pipe" from another window to unblock
Opening pipe in blocking mode
Reading 1 byte from pipe
```

```
echo 'hi' > pipe
```

```
Read 1 bytes, first byte was h
```

```
#!/bin/sh
make
rm pipe
mkfifo pipe
echo "Attempting read on a pipe, will block"
echo "Run \"echo 'hi' > pipe\" from another window to unblock"
./readfile pipe
```

```
aesd@aesd-VirtualBox:~/aesd-lectures/lecture5$ ./run-pipe-read-nonblocking.sh
make: Nothing to be done for 'all'.
Attempting read on a pipe in non-blocking mode

Using nonblocking mode as specified by argument
Opening pipe in nonblocking mode
Reading 1 byte from pipe
EAGAIN returned from read(), no data available to read
```

```
#!/bin/sh
make
rm -f pipe
mkfifo pipe
# Open a command for reading the pipe and put in the bac
# This prevents the next command from blocking
# See https://unix.stackexchange.com/questions/366219/pr
echo < pipe &
# Open a command to write to the pipe, but don't write a
# EOF
exec 3>pipe
echo "Attempting read on a pipe in non-blocking mode"
./readfile pipe nonblock
```

Linux System Programming Chapter 2
https://github.com/cu-ecen-5013/aesd-lectures/blob/master/lecture5

Writing Files

- write()

```
WRITE(2)                                                Linux Programmer's Manual

NAME
       write - write to a file descriptor

SYNOPSIS
       #include <unistd.h>

       ssize_t write(int fd, const void *buf, size_t count);

DESCRIPTION
       write() writes up to count bytes from the buffer starting at buf to the file referred to by the file descriptor fd.
```

Synchronized I/O

- Is data written to disk when write() completes successfully?
  - No - the write is into a kernel buffer.
- Why?
  - Disks are slow!

Synchronized I/O

- When is this a problem?
  - Unclean shutdown
- How can we manage the risk?
  - fsync(), fdatasync(), O_SYNC
  - Configure disk write caching.
- Avoid sync() for performance reasons