

O_TRUNC 这个截断是什么意思？

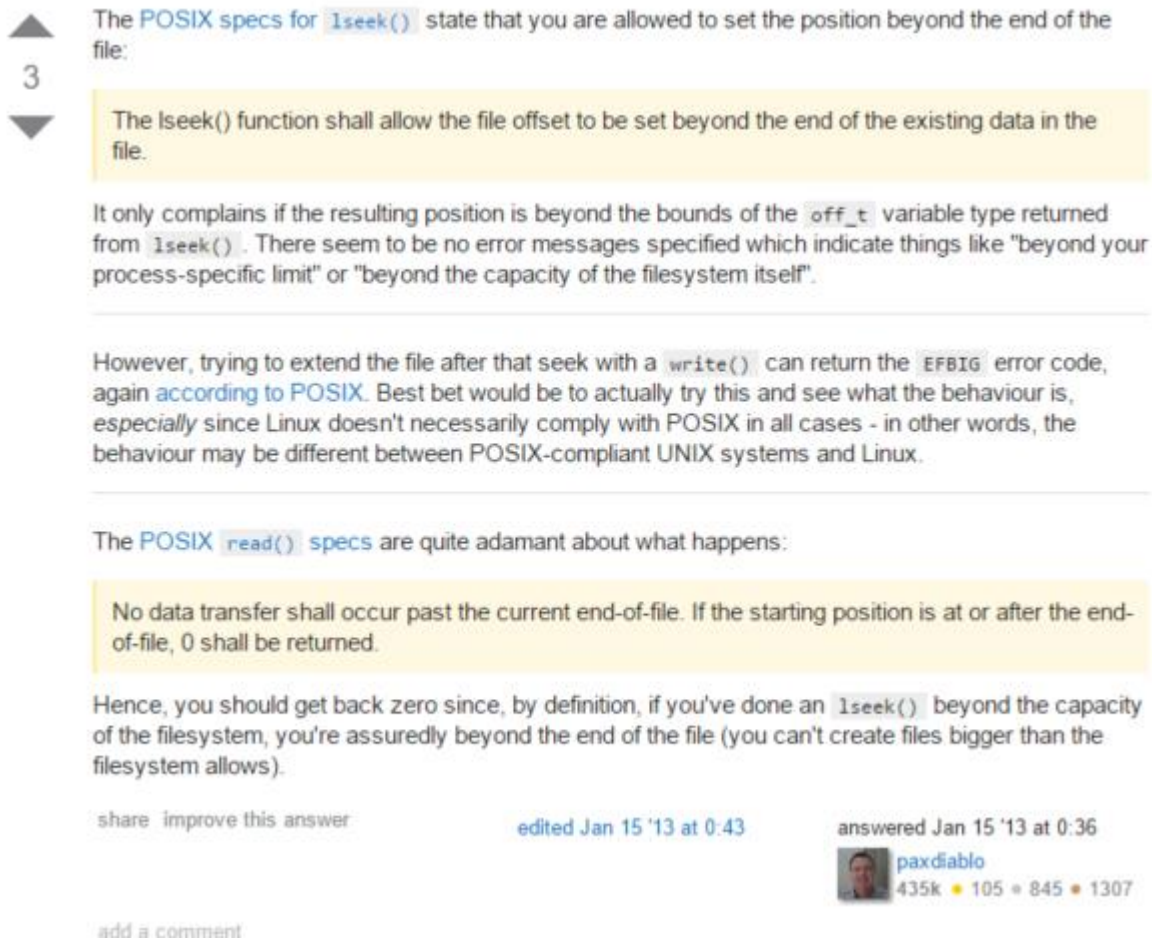
就是把文件的长度变为 0，原有数据清空的意思。

缺页异常时如何找到对应磁盘上的位置？

Linux 内核除了页表，还对每个进程维护一个虚拟内存的映射关系表。其中就包括这段区域映射到哪个文件、偏移量多少的信息。可以访问 <http://stackoverflow.com/questions/20041212/does-virtual-memory-area-struct-only-comes-into-picture-when-there-is-a-page-fault> 增进理解。然而这些并不在要求掌握的范围内。

如果 lseek 指定指针到文件末尾之外，那么读写会发生什么现象呢？

<http://stackoverflow.com/questions/14329187/unix-lseek-read-write-behavior>



The POSIX specs for `lseek()` state that you are allowed to set the position beyond the end of the file:

The `lseek()` function shall allow the file offset to be set beyond the end of the existing data in the file.

It only complains if the resulting position is beyond the bounds of the `off_t` variable type returned from `lseek()`. There seem to be no error messages specified which indicate things like "beyond your process-specific limit" or "beyond the capacity of the filesystem itself".

However, trying to extend the file after that seek with a `write()` can return the `EFBIG` error code, again according to POSIX. Best bet would be to actually try this and see what the behaviour is, especially since Linux doesn't necessarily comply with POSIX in all cases - in other words, the behaviour may be different between POSIX-compliant UNIX systems and Linux.

The POSIX `read()` specs are quite adamant about what happens:

No data transfer shall occur past the current end-of-file. If the starting position is at or after the end-of-file, 0 shall be returned.

Hence, you should get back zero since, by definition, if you've done an `lseek()` beyond the capacity of the filesystem, you're assuredly beyond the end of the file (you can't create files bigger than the filesystem allows).

share improve this answer edited Jan 15 '13 at 0:43 answered Jan 15 '13 at 0:36 paxdiablo 435k 105 845 1307 add a comment

要多多运用搜索引擎哦。

关于标准错误，是不是也是一种输出？

默认情况下，标准输出和标准错误都会输出到控制台。是的，它也是一种输出文件。

关于 v-node 的解释

Linux 有一种对文件系统的抽象，这就是所谓的 vfs（虚拟文件系统）。Vnode 就是这个文件系统节点，你可以理解为文件系统为了在内存中记录文件的元信息（metadata）而实现的结构。

另外，vnode 结构体中有一个引用计数。所以它同样可以追踪自己被引用多少次，并在恰当的时候释放掉。

```
typedef struct vnode {
    kmutex_t      v_lock;           /* protects vnode fields */
    u_short       v_flag;           /* vnode flags (see below) */
    u_long        v_count;          /* reference count */
    struct vfs     *v_vfsmountedhere; /* ptr to vfs mounted here */
    struct vnodeops *v_op;           /* vnode operations */
    struct vfs     *v_vfsp;          /* ptr to containing VFS */
    struct stdata  *v_stream;        /* associated stream */
    struct page    *v_pages;         /* vnode pages list */
    enum vtype     v_type;           /* vnode type */
    dev_t         v_rdev;           /* device (VCHR, VBLK) */
    caddr_t       v_data;           /* private data for fs */
    struct filock  *v_filocks;       /* ptr to filock list */
    kcondvar_t     v_cv;            /* synchronize locking */
} vnode_t;
```

可以参考 <http://everything2.com/title/vnode>

系统级 IO 课件上的思考题

【答案刮开复制可见】

Fun with File Descriptors (1)

```
#include "csapp.h"
int main(int argc, char *argv[])
{
    int fd1, fd2, fd3;
    char c1, c2, c3;
    char *fname = argv[1];
    fd1 = Open(fname, O_RDONLY, 0);
    fd2 = Open(fname, O_RDONLY, 0);
    fd3 = Open(fname, O_RDONLY, 0);
    Dup2(fd2, fd3);
    Read(fd1, &c1, 1);
    Read(fd2, &c2, 1);
    Read(fd3, &c3, 1);
    printf("c1 = %c, c2 = %c, c3 = %c\n", c1, c2, c3);
    return 0;
}                                     ffiles1.c
```

- What would this program print for file containing “abcde”?

答案是

：

Fun with File Descriptors (2)

```
#include "csapp.h"
int main(int argc, char *argv[])
{
    int fd1;
    int s = getpid() & 0x1;
    char c1, c2;
    char *fname = argv[1];
    fd1 = Open(fname, O_RDONLY, 0);
    Read(fd1, &c1, 1);
    if (fork()) { /* Parent */
        sleep(s);
        Read(fd1, &c2, 1);
        printf("Parent: c1 = %c, c2 = %c\n", c1, c2);
    } else { /* Child */
        sleep(1-s);
        Read(fd1, &c2, 1);
        printf("Child: c1 = %c, c2 = %c\n", c1, c2);
    }
    return 0;
}
```

ffiles2.c

- What would this program print for file containing "abcde"?

24

答案是

Parent: c1 = a, c2 = b
Child: c1 = b, c2 = c

Fun with File Descriptors (3)

```
#include "csapp.h"
int main(int argc, char *argv[])
{
    int fd1, fd2, fd3;
    char *fname = argv[1];
    fd1 = Open(fname, O_CREAT|O_TRUNC|O_RDWR, S_IRUSR|S_IWUSR);
    Write(fd1, "pqrs", 4);
    fd3 = Open(fname, O_APPEND|O_WRONLY, 0);
    Write(fd3, "jklmn", 5);
    fd2 = dup(fd1); /* Allocates descriptor */
    Write(fd2, "wxyz", 4);
    Write(fd3, "ef", 2);
    return 0;
}
```

ffiles3.c

- What would be the contents of the resulting file?

25

答案是

efpqrsjklmnwxyz

一些有趣的程序片段

【同样地，用力刮开黑色块可以看到答案】

Forker

```
void handler(int sig) {
    printf("s");
}
```

```

        exit(7);
    }

int forker() {
    pid_t pid;
    int state;
    signal(SIGINT, handler);

    printf("A");
    fflush(stdout);

    pid = fork();
    printf("B");
    if (pid == 0)
        printf("C");
    else {
        kill(pid, SIGINT);
        waitpid(pid, &state, 0);
        printf("%d", WEXITSTATUS(state));
    }
    printf("E");
    exit(4);
}

```

输出可能是什么呢？（多选）

ABBCE4E ABs7E ABCEsB4E AB4EBCE ABCBs7E ABCEB4sE

答案：

☐

☐

☐

☐

Fdplay

```

#include "csapp.h"

int fdplay() {
    int pid;
    int fd1, fd2;

    fd1 = open("file1", O_RDWR);
    dup2(fd1, 1);
    printf("A");
    if ((pid = fork()) == 0) {
        printf("B");
        fd2 = open("file1", O_RDWR);
        dup2(fd2, 1);
    }
}

```

```

        printf("C");
    }
    else {
        waitpid(pid, NULL, 0);
        printf("D");
        close(fd1);
        printf("E");
    }
    exit(2);
}

int main() {
    fdplay();
    return 0;
}

```

如果一开始的时候，file1 中为空，则上面这个会使得 file1 中有什么？

答案：

```


```

```


```

```


```

```


```

```


```

```


```

```


```

```


```

```

#include "csapp.h"

int fdplay() {
    int pid;
    int fd1, fd2;

    fd1 = open("file1", O_RDWR | O_TRUNC | O_CREAT, S_IRUSR | S_IWUSR);
    dup2(fd1, 1);
    printf("A");
    if ((pid = fork()) == 0) {
        printf("B");
        fd2 = open("file1", O_WRONLY);
        system("cat file1 > capture");
        dup2(fd2, 1);
        system("cat file1 > capture2");
        printf("C");
    }
    else {
        waitpid(pid, NULL, 0);
        system("cat file1 > capture3");
        printf("D");
    }
}

```

```
        close(fd1);  
        printf("E");  
    }  
    exit(2);  
}  
int main() {  
    fdplay();  
    return 0;  
}
```