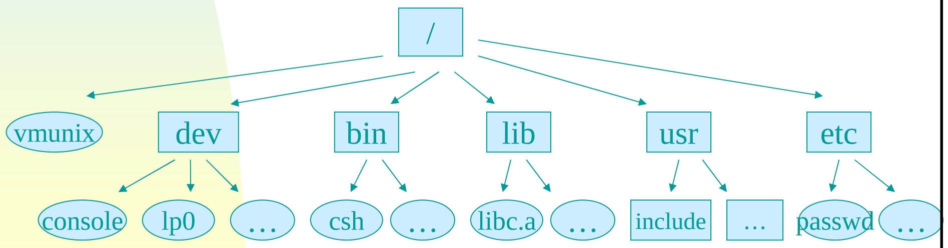
Contents

File I/O

- Objective of this chapter:
  - Functions available for file I/O
    - POSIX.1 & XPG3, instead of ANSI C
  - Atomic operations in multiprogramming environments
- Unbuffered I/O
  - Popular functions: open, close, read, write, lseek, dup, fcntl, ioctl
  - Each read() and write() invokes a system call!

File I/O

- File
  - A sequence of bytes
- Directory
  - A file that includes info on how to find other files.



File I/O

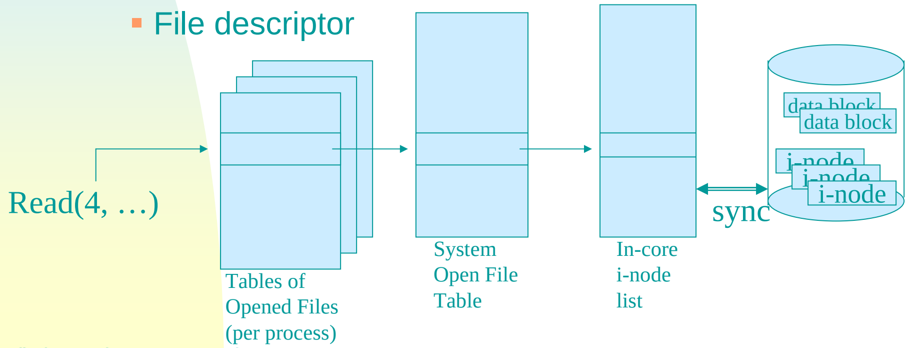
- Path name
  - Absolute path name
    - Start at the root / of the file system
    - /user/john/fileA
  - Relative path name
    - Start at the “current directory” which is an attribute of the process accessing the path name.
    - ./dirA/fileB
- Links
  - Symbolic Link – 4.3BSD
    - A file containing the path name of another file can across file-system boundaries.
  - Hard Link
    - . or ..

File I/O

- File Descriptor
  - Non-negative integer returned by open() or creat(): 0 .. OPEN_MAX
    - Virtually un-bounded for SVR4 & 4.3+BSD
  - Per-process base
  - POSIX.1 – 0: STDIN_FILENO, 1: STDOUT_FILENO, 2: STDERR_FILENO
    - <unistd.h>
    - Convention employed by the Unix shells and applications

File I/O - File Manipulation

- Operations
  - open, close, read, write, lseek, dup, fcntl, ioctl, trunc, rename, chmod, chown, mkdir, cd, opendir, readdir, closedir, etc.
    - File descriptor



File I/O – open

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

- O_ RDONLY, O_WRONLY, O_RDWR
- O_APPEND, O_TRUNC, O_NOCTTY
- O_CREAT, O_EXCL
- O_NONBLOCK, O_SYNC
- File/Path Name
  - PATH_MAX, NAME_MAX
  - _POSIX_NO_TRUNC -> ENAMETOOLONG if error occurs (NAME_MAX or PATH_MAX).

File I/O – creat and close

#include <sys/types>
#include <sys/stat.h>
#include <fcntl.h>
int creat(const char*pathname, mode_t mode);

- open(pathname, O_WRONLY | O_CREAT | O_TRUNC, mode)
- Only for write-access

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

- All open files are automatically closed by the kernel when a process terminates.

File I/O - lseek

#include <sys/types>

#include <unistd.h>

off_t lseek(int filedes, off_t offset, int whence);

- Current file offset in bytes
- whence: SEEK_SET, SEEK_CUR, SEEK_END
- Example

  currpos = lseek(fd, 0, SEEK_CUR)
    - EPIPE for a pipe or a FIFO
- off_t: typedef long   off_t;   /* 2^{31} bytes */
  - or typedef longlong_t   off_t;   /* 2^{63} bytes */
  - Negative for /dev/kmem on SVR4

File I/O - lseek

- Program 3.1 – Page 52
  - Test if "standard input" is capable of seeking?
  - cat < /etc/motd | a.out → cannot seek a FIFO or pipe (EPIPE)
  - a.out < /var/spool/cron/FIFO
- Program 3.2 – Page 53, hole creating!
  - od –c file.hole → 000000 a b c \0 \0 \n

    000006

File I/O – read and write

#include <unistd.h>

ssize_t read(int filedes, void *buf, size_t nbytes);

- Less than nbytes of data are read:
  - EOF, terminal device (line-input), network buffering, record-oriented devices (e.g., tape)
  - Offset is increased for every read() – SSIZE_MAX

#include <unistd.h>

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

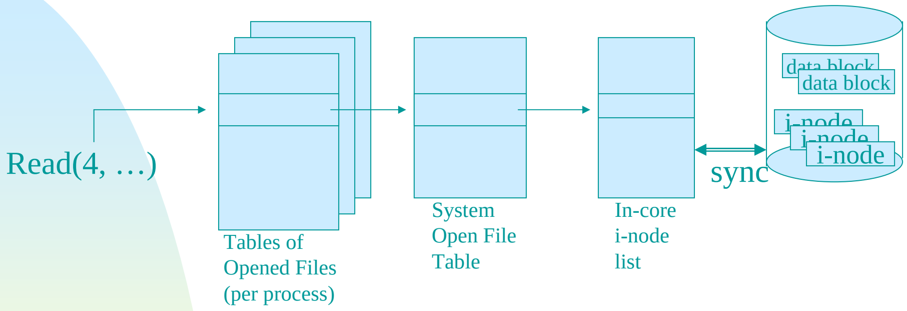
  - Write errors for disk-full or file-size-limit causes.
  - O_APPEND

File I/O - Efficiency

- Program 3.3 – Page 56
  - No needs to open/close standard input/output
  - Copy stdin to stdout (> /dev/null)
  - Try I/O redirection in reading an 1.4M file

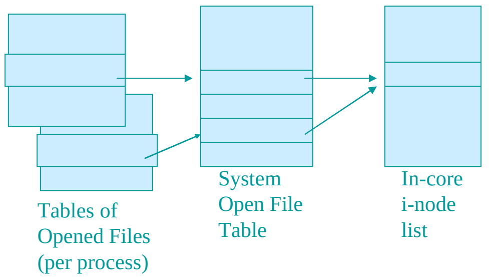
Buffersize	UsrCPU	SysCPU	Clock	#loops
1	23.8s	397.9s	423.4s	1468802
64	0.3s	6.6s	7.0s	22951
512	0.0s	1.0s	1.1s	2869
1024	0.0s	0.6s	0.6s	1435
8192	0.0s	0.3s	0.3s	180
131072	0.0s	0.3s	0.3s	12

File I/O – Sharing



- Table per process: filedes flags, a pointer
- Sys open file table: file status, offset, a v-node pointer
- V-node (since 4.3BSD Reno)
  - i-node: owner, file size, residing device, block ptr,..
    - In SVR4, i-node contains/is replaced with v-node.
  - Peter Weinberger (Bell Lab)/Bill Joy (Sun)

File I/O – Sharing



- Each “independently opened file” has its offset.
- Examples
  - Write → offset is incremented!
  - O_APPEND → offset = current file size before each write
  - lseek() causes no I/O (only on the system open file table)
  - dup() and fork causes the sharing of entries in the (system open) file table.
  - filedes flags versus file status flags

File I/O – Atomic Operations

- Atomic Operation
  - Composed of multiple steps?
- Example – File Appending

```
if (lseek(fd, 0L, 2) < 0 err_sys(“lseek err”);
if (write(fd, buf, 10) != 10) err_sys(“wr err);
```

- Example – File Creation

```
if ((fd=open(pathname, O_WRONLY)) < 0)
   if (errno == ENOENT) {
     if ((fd = creat(pathname, mode)) < 0)
        err_sys(“creat err”);
   } else err_sys(“open err);
```

  - creat() rewrites and truncates any existing file.

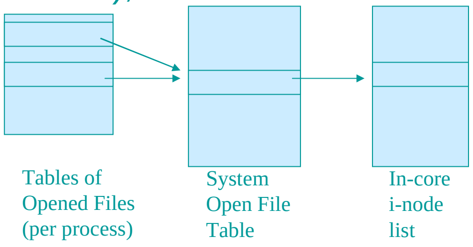
File I/O – dup and dup2

#include <unistd.h>

int dup(int filedes);

int dup2(int filedes, int newfiledes);

- dup() returned the lowest available filedes.
- dup2() is atomic and from Version 7, …SVR3.2
  - close(newfiledes); fcntl(fildes, F_DUPFD, newfiledes);



File I/O - fcntl

```
#include <sys/types>
#include <unistd.h>
#include <fcntl.h>
int fcntl(int filedes, int cmd, … /* int arg */);
```

- Changes the properties of opened files
- F_DUPFD: duplicate an existing file descriptor (>= arg).
  - FD_CLOEXEC is cleared (for exec()).
- F_GETFD, F_SETFD: filedes flag, e.g., FD_CLOEXEC
- F_GETFL, F_SETFL: file status flags
  - O_APPEND, O_NOBLOCK, O_SYNC, O_ASYNC, O_RDONLY, O_WRONLY, RDWR
- F_GETOWN, F_SETOWN: ownership, + procID, -groupID
  - SIGIO, SIGURG – I/O possible on a filedes/urgent condition on I/O channel

File I/O - fcntl

- Program 3.4 – Page 65
  - Print file flags for a specified descriptor
- Program 3.5 – Page 66
  - Turn on one or more flags
    - val &= ~flags
    - set_fl(STDOUT_FILENO, O_SYNC);
- O_SYNC writes

Operation	UsrCPU	SysCPU	Clock
Async, > /dev/null	0.0s	0.3s	0.3s
Async, > disk file	0.0s	1.0s	2.3s
Sync, > disk file	0.0s	1.4s	13.4s

File I/O - ioctl

#include <unistd.h>

#include <sys/ioctl.h>

int ioctl(int filedes, int request);

- Catchall for I/O operations – not in POSIX.1
  - E.g., setting of the size of a terminal's window.
- SVR4 prototype
- More headers could be required:
  - Disk labels (<disklabel.h>), file I/O (<ioctl.h>), mag tape (<mtio.h>), socket I/O (<ioctl.h>), terminal I/O (<ioctl.h>)

File I/O - /dev/fd

- /dev/fd/n
  - open("/dev/df/*n*",mode) → duplicate descriptor *n* (assuming that *n* is open)
    - open("/dev/df/0",mode) == fd=dup(0)
    - The new mode is a subset of that of the referenced file.
  - Uniformity and Cleanliness!
    - Not POSIX.1, but supported by SVR4 and 4.3+BSD
      - /dev/stdin -> ./fd/0
    - cat /dev/fd/0