

Computer Programming

Lecture 2

이윤진
서울대학교
2007.12.24.

Slide Credits

- 엄현상 교수님
 - 서울대학교 컴퓨터공학부
 - Computer Programming, 2007 봄학기

Unix/Linux (3)

순서

- File system
 - 지난 시간 정리
 - Sample file system
 - Link, unlink, remove, and rename
 - File descriptors, file table, and inode table
- Process control
 - Command in the shell
 - Process control primitives
 - Examples
 - Booting & logging-in
 - 환경변수 설정

File System

File System 정리

- File system
 - Abstraction Used by the Kernel to Represent and Organize the System's Storage Resources
- File system의 특징
 - Hierarchical Structure
 - Consistent Treatment of File Data
 - Ability to Create and Delete Files
 - Dynamic Growth of Files
 - Protection of File Data
 - Treatment of Peripheral Devices as Files

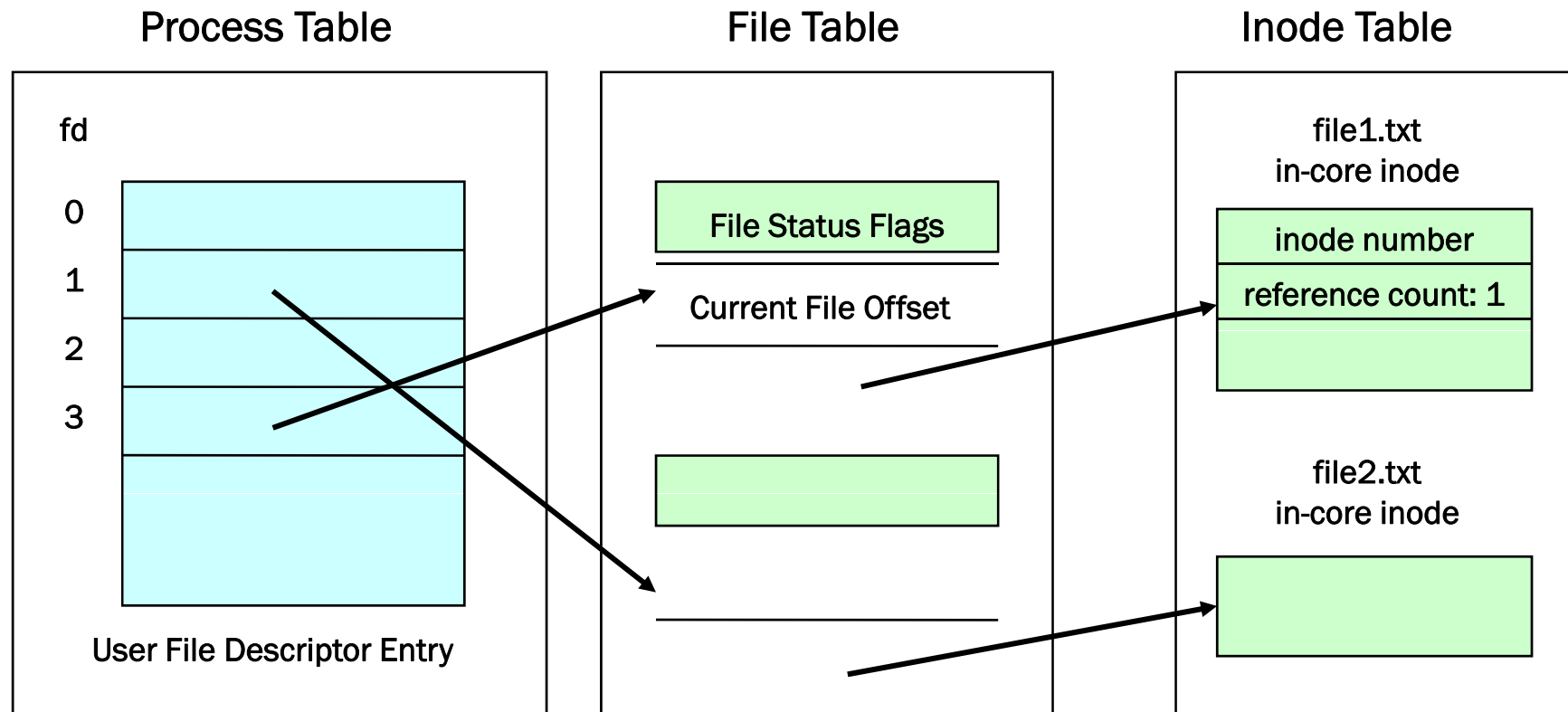
File System 정리 (계속)

- Inode (Static on Disk)
 - Information about each file in a structure that the kernel maintains
 - Owner of the file
 - Size of the file
 - Device that the file is located on
 - Pointers to where the actual data blocks for the file are located on disk
 - Type of the file
 - Access permission of the file
 - Access times of the file
 - (Hard) Link Count/Number

File Descriptors, File & Inode Tables

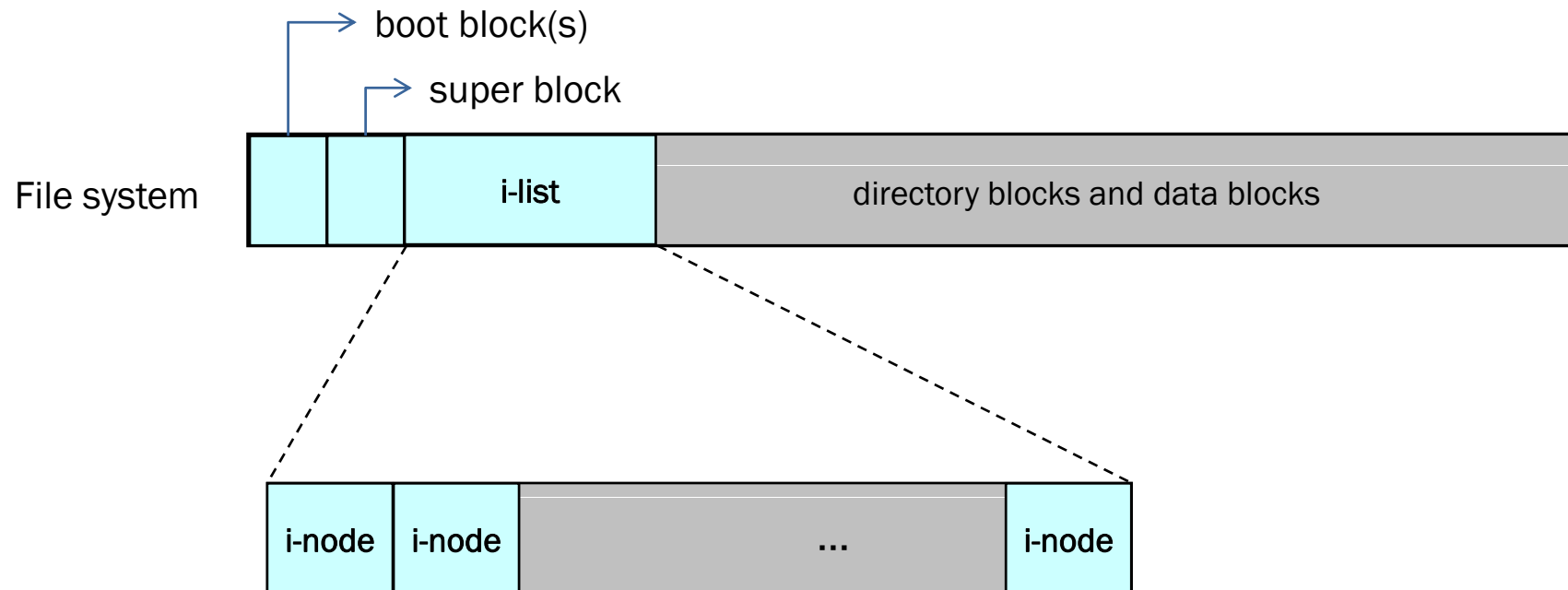
- Kernel Data Structure for Open Files

```
martini:~> cat file1.txt > file2.txt
```

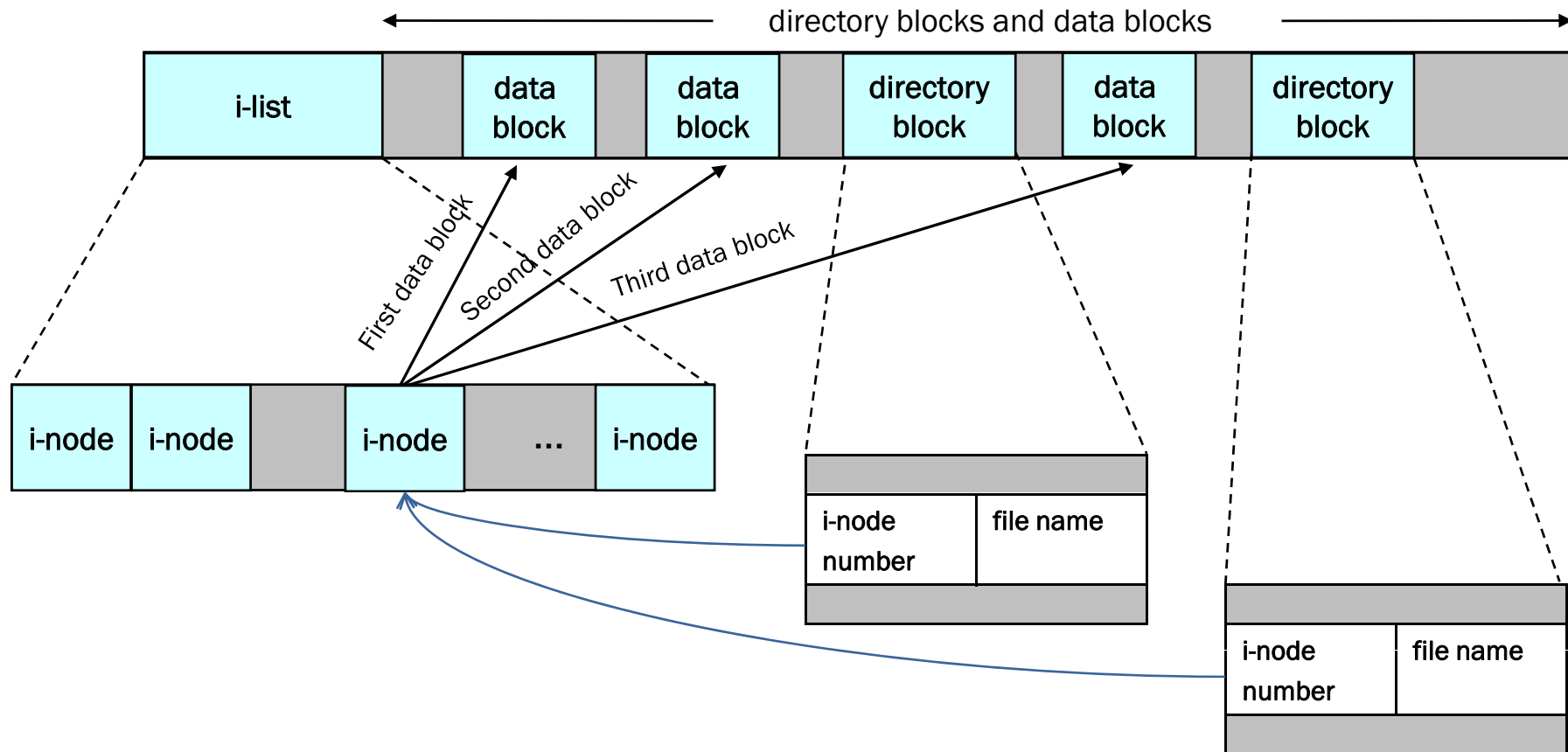


File System Layout

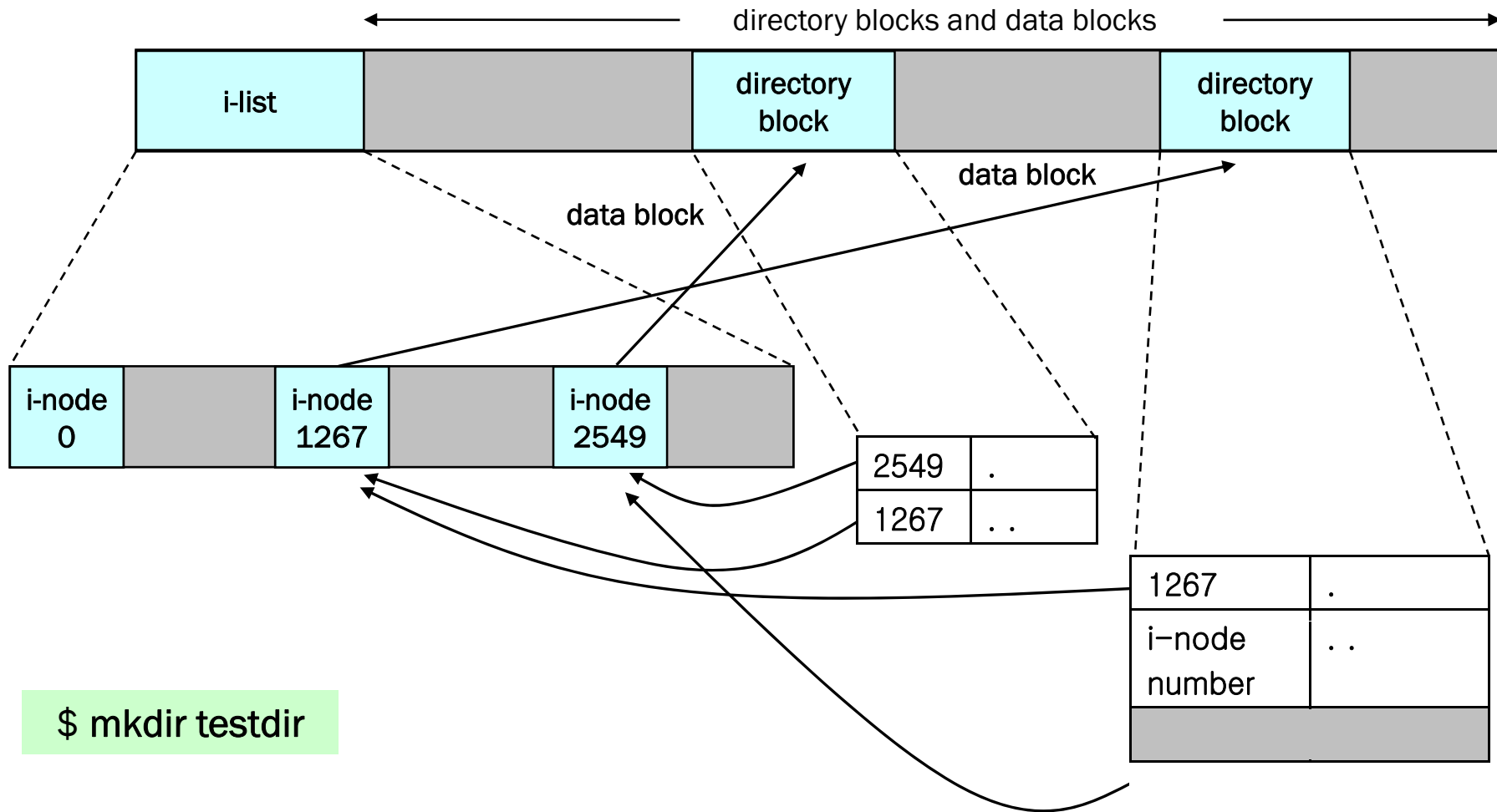
- Boot Block: OS가 부팅될 수 있도록 하는 boot sector
- Super Block: file system의 크기, block 개수, inode 개수, 각종 날짜, file system magic number 등 file system을 인식하는데 필요한 정보 저장



File System Layout in More Detail



Sample File System



Link, Unlink, Remove and Rename

- Link and Unlink

```
int link(const char *existingpath, const char *newpath);  
/* create a new directory entry, newpath, that references the  
existing file existingpath. */  
int unlink(char *pathname);  
/* remove the directory entry and decrements the link count */
```

- Remove

```
int remove(const char *pathname);
```

- Rename

```
int rename(const char *oldname, const char *newname);
```

Unlink Code Example

```
#include <fcntl.h>

int main (void)
{
    if(open("tempfile", O_RDWR) < 0)
        printf("open error\n");
    if(unlink("tempfile") < 0)
        printf("unlink error\n");
    printf("file unlinked\n");
    sleep(15);
    printf("done\n");
    exit(0);
}
```

Unlink Code Example (계속)

- Result

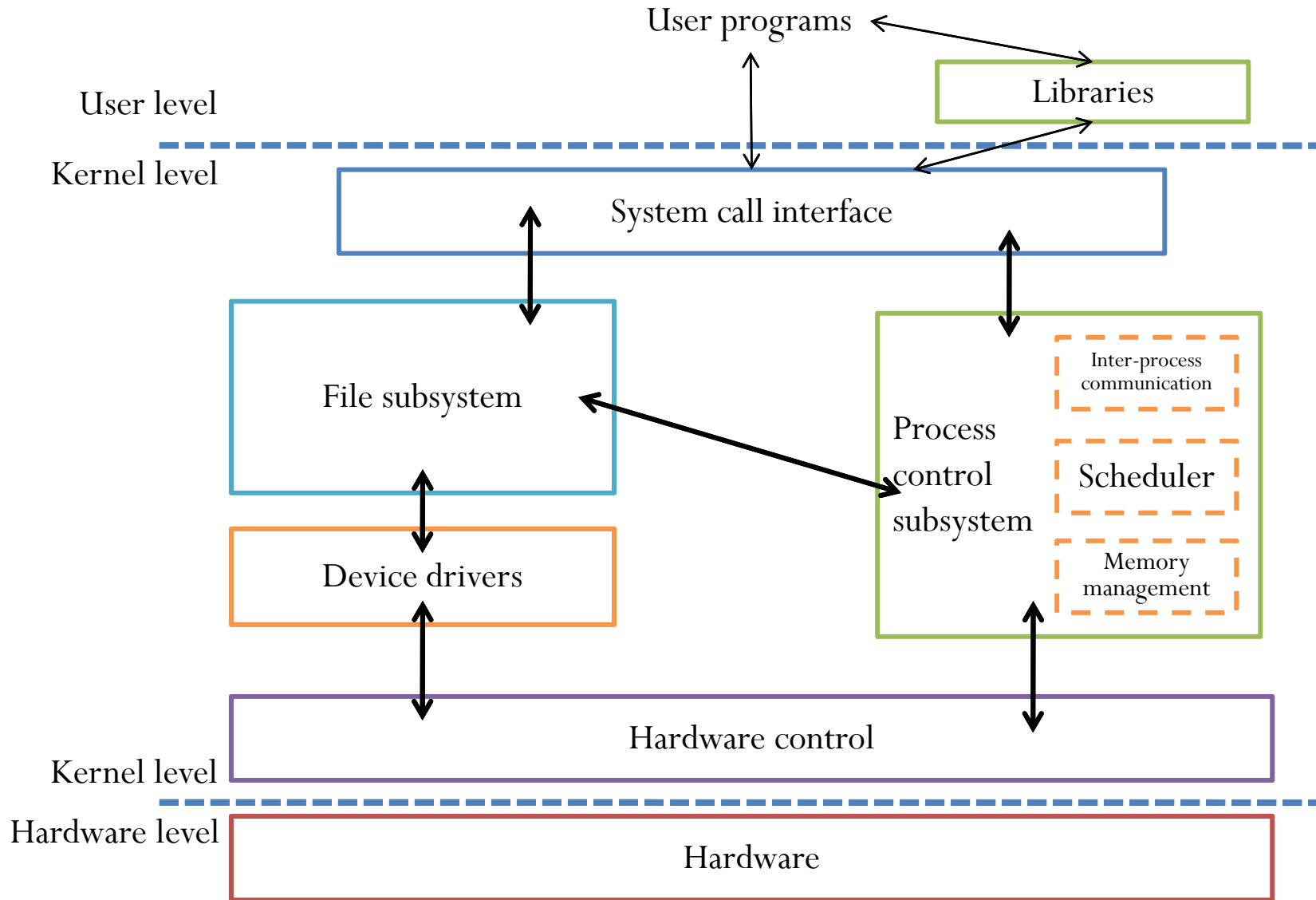
```
shell [ jin{63} ~] ls tempfile
0 tempfile
shell [ jin{64} ~] a.out
file unlinked
done
shell [ jin{65} ~] ls tempfile
ls: tempfile: No such file or directory
shell [ jin{66} ~]
```

- Relevant function

```
unsigned sleep(unsigned seconds);
/* delay for a specific amount of time */
```

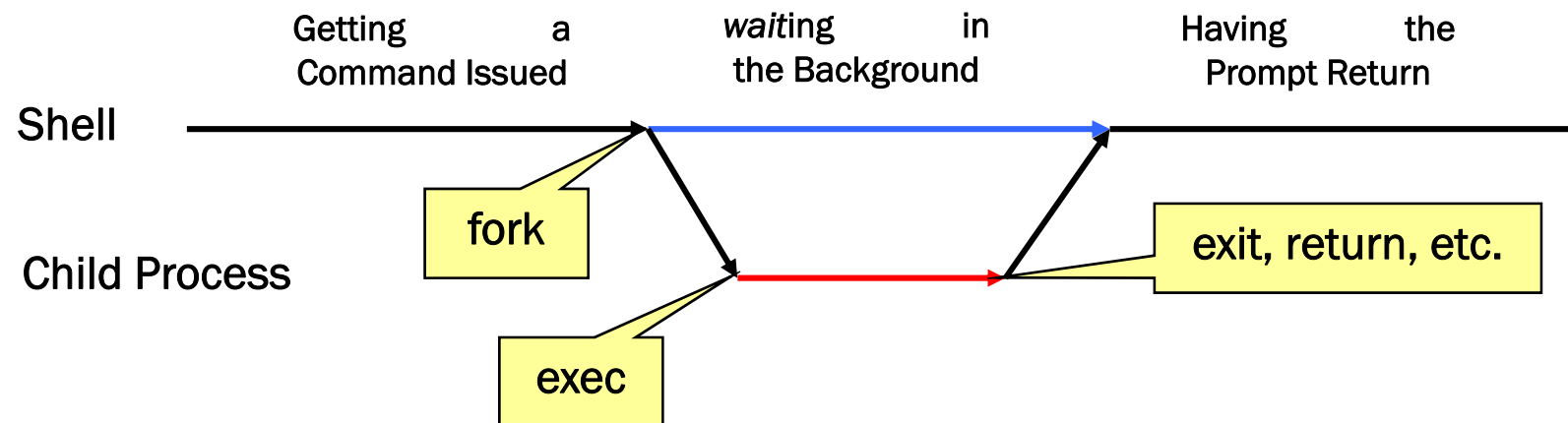
Process Control

UNIX Kernel



Command in the Shell

- Issue of a Command at the Shell Level
 - Generate a Child Process (by Calling *fork*)
 - Replace the Current (Shell) Image with the Command Image (by Calling *exec*)
- Execution of a Command via Process Creation



Process Control Primitives

- fork
 - Only Way for an Existing Process to Create a New Process (Its Copy)
 - Return 0 in child
 - Return process ID in parent
 - Return -1 on error

```
#include <sys/types.h>  
pid_t fork(void);
```

- Text Segment Shared If It Is Read-Only
 - Copy-On-Write (COW)
 - Parent's Data Space, Stack, and Heap Shared Until Being Modified
 - Copied at the modification time

Process Control Primitives (계속)

- exec
 - Ways to Execute another Program
 - Return -1 on error
 - No return on success
 - Current Process Replaced by a New Program on Disk
 - Six Different Versions
 - execl, etc.

```
int execl(const char *pathname, const char *arg0, ... /* (char *) 0 */ );
```

Process Control Primitives (계속)

- wait
 - Way to Block the Caller Until a Child Process Terminates

```
#include <sys/types.h>  
pid_t wait(int *statloc);
```

- exit
 - Ways for a Process to Terminate Normally
 - exit status
 - argument to exit or _exit as the return value from main

```
void exit(int status);
```

Process Control Code Example

```
#include <sys/types.h>
main(){
    int status;
    pid_t pid;
    if(fork() == 0){
        /* Child process */
        pid = getpid();
        printf("Child process' PID = %d \n", pid);
        sleep(10);
        exit(0);
    } else{
        /* Parent process*/
        pid = getpid();
        printf(" \nParent process' PID = %d \n", pid);
        wait(&status);
    }
    exit(0);
}
```

Process Control Code Example

- Result

```
martini:~> gcc -o fork fork.c
martini:~> fork &
[1] 6366
martini:~>
Parent process' PID = 6366
Child process' PID = 6367
martini:~> ps
  PID TTY          TIME CMD
 5361 pts/2    00:00:00 bash
 6366 pts/2    00:00:00 fork
 6367 pts/2    00:00:00 fork
 6368 pts/2    00:00:00 ps
martini:~> [Enter]
[1]  Done                fork
```

Process Control

Command Execution Example

- Command Execution Example

```
martini:~>cat file1.txt
123
martini:~> cat file1.txt > file2.txt
martini:~>cat file2.txt
123
martini:~> cat file2.txt > /dev/tty
```

Current Terminal

- Result

```
123
```

```
martini:~> exec cat file1.txt > file2.txt
```

- Result

- ???

Process Control의 시작

- Booting (Bootstrapping)

: Start Up a Computer

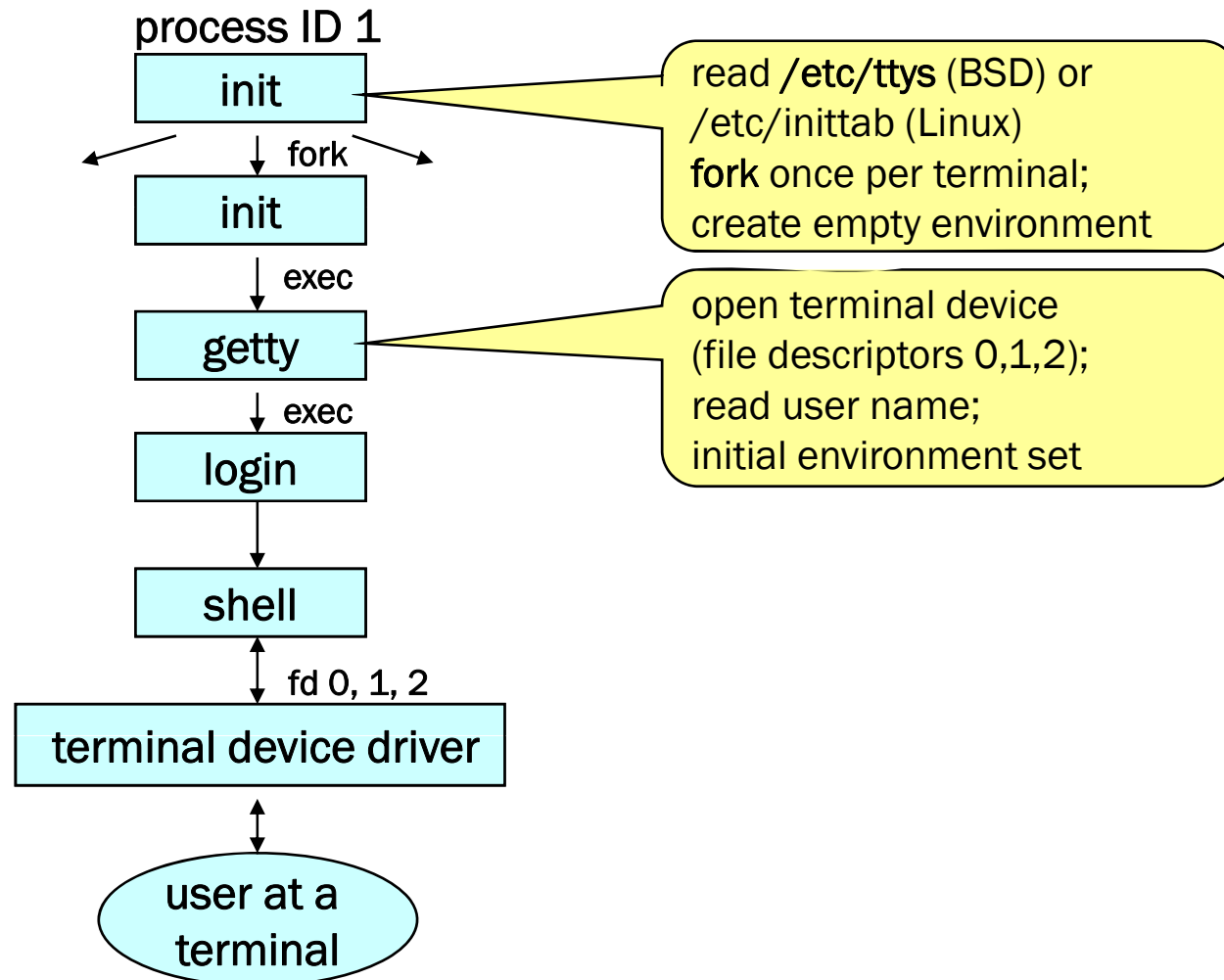
- Load the Kernel into Memory
- Perform Various Initialization Tasks
 - init process (/sbin/init) invoked at the end
 - getty program executed per terminal

- Logging in to the System

: Invoke the login Process

- Initialize the Environment
- Read the Start-up Files
 - .bash_profile, .bash_login, & .bashrc or .login & .cshrc
- Change/Add Some Environment Variables

Booting & Logging-In Processes



환경변수 설정

- Environment Variables

```
martini:~>env  
USER=net001  
LOGNAME=net001  
HOME=/csehome/net001  
PATH=/usr/local/bin:/usr/bin:/bin  
SHELL=/bin/bash  
: (omitted)
```

```
martini:~>export PATH=$PATH:.
```

```
martini:~>set PATH = ($PATH .)
```



csh

- Result

```
martini:~>env  
:  
PATH=/usr/local/bin:/usr/bin:/bin:.
```

Unix/Linux (4)

순서

- Process Control
 - Interprocess Communication
 - Miscellaneous
 - Example

Interprocess Communication (IPC)

- Pipes
 - Mechanisms That Allow a Stream of Data to be Passed Between Two Related Processes
- FIFOs
 - Mechanisms That Allow Data to be Exchanged by Unrelated Processes
- Message Queues
 - Linked List of Messages Stored within the Kernel and Identified by a Message Queue Identifier
- Semaphores
 - Counters Used to Provide Access to a Shared Data for Multiple Processes
- Shared Memory
 - Mechanism that Allows Multiple Processes to Share a Given Region of Memory
- Sockets
 - Mechanism that Allows (TCP/IP) Network Communication between Processes

Interprocess Communication (계속)

- fork
 - Check for Available Kernel Resources
 - Free Process Table Entry
 - Copy Data from Parent Process Table Entry to New Child Entry
 - Increment Counts
 - Copy Parent Program in Memory
 - Return Child Process ID in Parent
 - Return 0 in Child
 - Return -1 on Error

Pipe Code Example

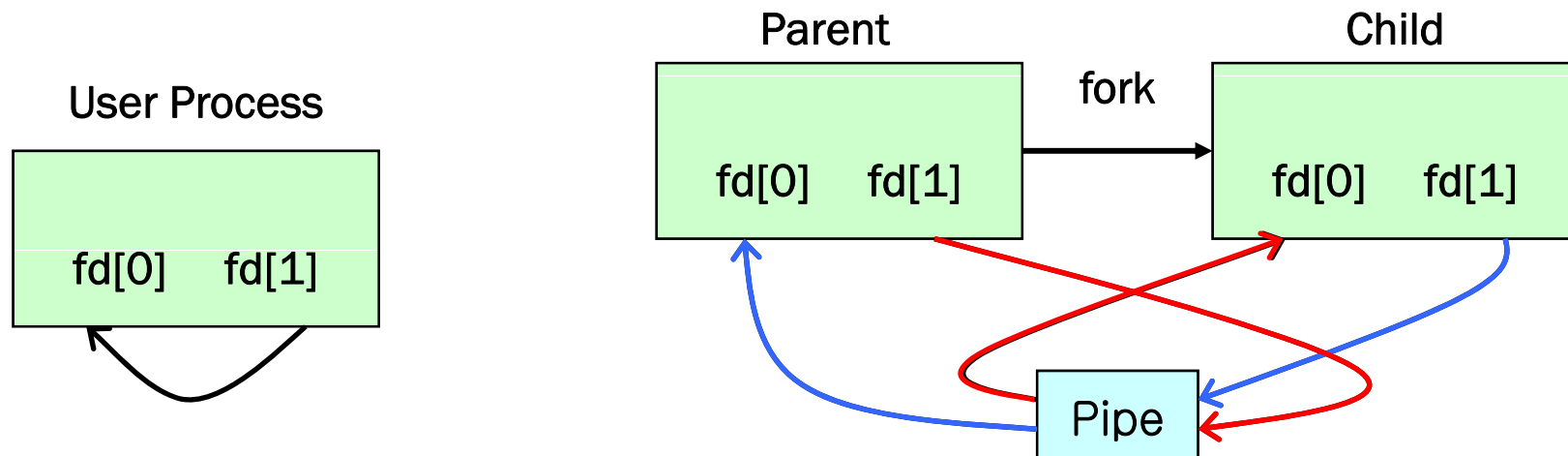
```
#include <sys/types.h>
main(){
    int n, fd[2];
    pid_t pid;
    char line[100];
    if (pipe(fd) < 0){
        printf("pipe error! \n");
        exit(1);
    }
    if ((pid = fork()) < 0) {
        printf("fork error! \n");
        exit(1);
    } else if (pid > 0) { /* Parent Process */
        close(fd[0]);
        write(fd[1], "hello world \n", 12);
    } else { /* Child Process */
        close(fd[1]);
        n = read(fd[0], line, 100);
        write(1, line, n);
    }
    exit(0);
}
```

Pipe Code Example (계속)

- Result

```
martini:~> gcc -o pipe pipe.c
martini:~> pipe
hello world
```

- Way to view a Pipe



Miscellaneous

- Signals
 - Technique Used to Notify a Process That Some Condition Has Occurred

```
#include <signal.h>
... (omitted)
signal(SIGINT, sig_int); // Handler Establishment
...
void sig_int(int signo) {
    printf("Interrupt \n");
}
```

- Software Interrupts

Miscellaneous (계속)

- Interrupts
 - Signals That Get the Attention of the CPU
 - Usually Made By Devices
 - Serviced Between Instructions
 - Prioritized
- Disk Usage

```
martini:~>du .  
20  ./Dir  
120 .  
martini:~>quota
```

FIFO & Signal Code Example

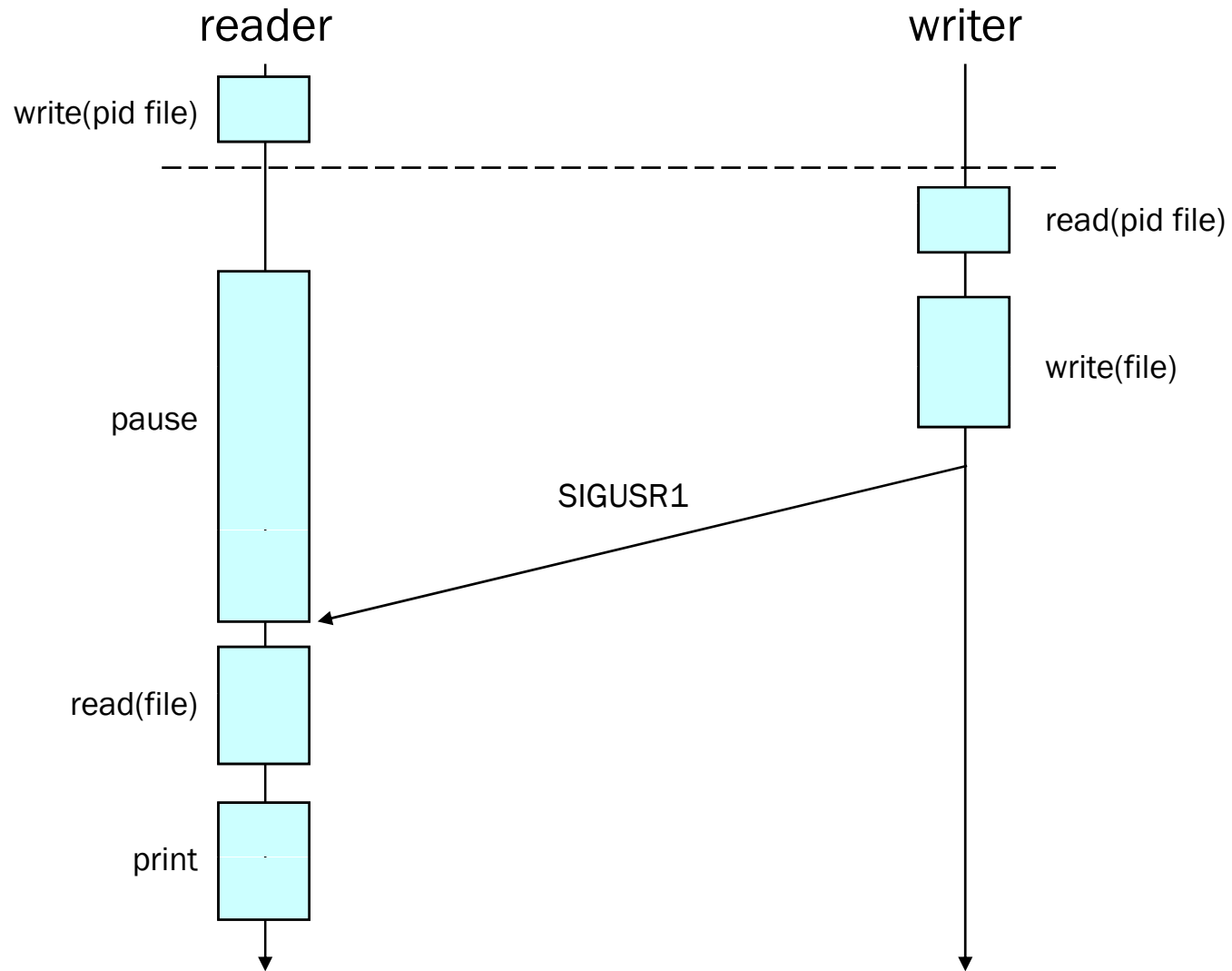
```
#include <sys/types.h>
#include <signal.h>
#include <sys/stat.h>
#include <fcntl.h>

char line[100];
int fd_fifo;

void sig_handler(int sig){
    printf("I'm a signal handler! \n");
    read(fd_fifo, line, 100);
    printf("%s", line);
}
```

```
main(){
    pid_t pid;
    signal(SIGUSR1, sig_handler);
    if((pid = fork()) > 0){
        // parent
        mkfifo("fifo", 0666);
        fd_fifo = open("fifo", O_WRONLY);
        write(fd_fifo, "hello, world! \n", 15);
        kill(pid, SIGUSR1);
        wait();
    } else{
        // child
        sleep(1);
        fd_fifo = open("fifo", O_RDONLY);
        pause();
    }
    unlink("fifo");
    exit(0);
}
```

IPC Example



IPC Code Example - Reader

```
#include <sys/types.h>
#include <signal.h>
#include <fcntl.h>
pid_t rPid;
int fileFd = 0;
void sig_handler(int sig){
    char message[100];
    fileFd = open("file", O_RDONLY);
    read(fileFd, message, 100);
    printf("%s \n",message);
}
main(){
    signal(SIGUSR1, sig_handler);
    /* write the reader's pid */
    if (!fileFd) /* if the signal has not arrived */
        pause();
    exit(0);
}
```

IPC Code Example - Writer

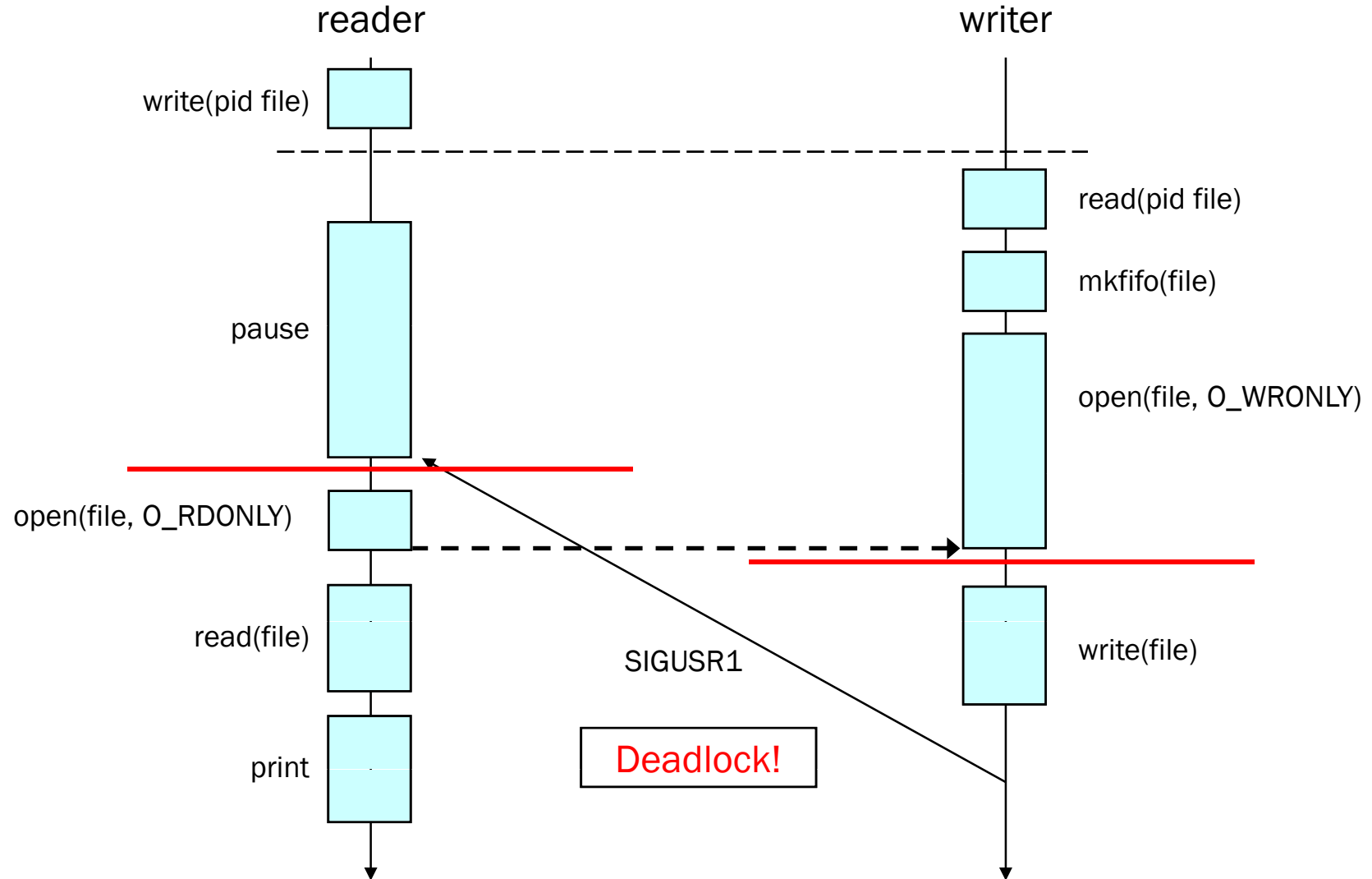
```
#include <sys/types.h>
#include <signal.h>
#include <fcntl.h>
char message[] = " \ nReader, Print This Message!";
pid_t rPid;
main(){
    int fileFd;
    /* if the reader's pid can be read, read it and set rPid to it;
       otherwise, sleep for a while and try again */
    creat("file",0666); /* if replaced by mkfifo("file", 0666);, what will happen??? */
    fileFd = open("file", O_WRONLY);
    write(fileFd, message, strlen(message)+1);
    kill(rPid, SIGUSR1);
    exit(0);
}
```

IPC Code Example (계속)

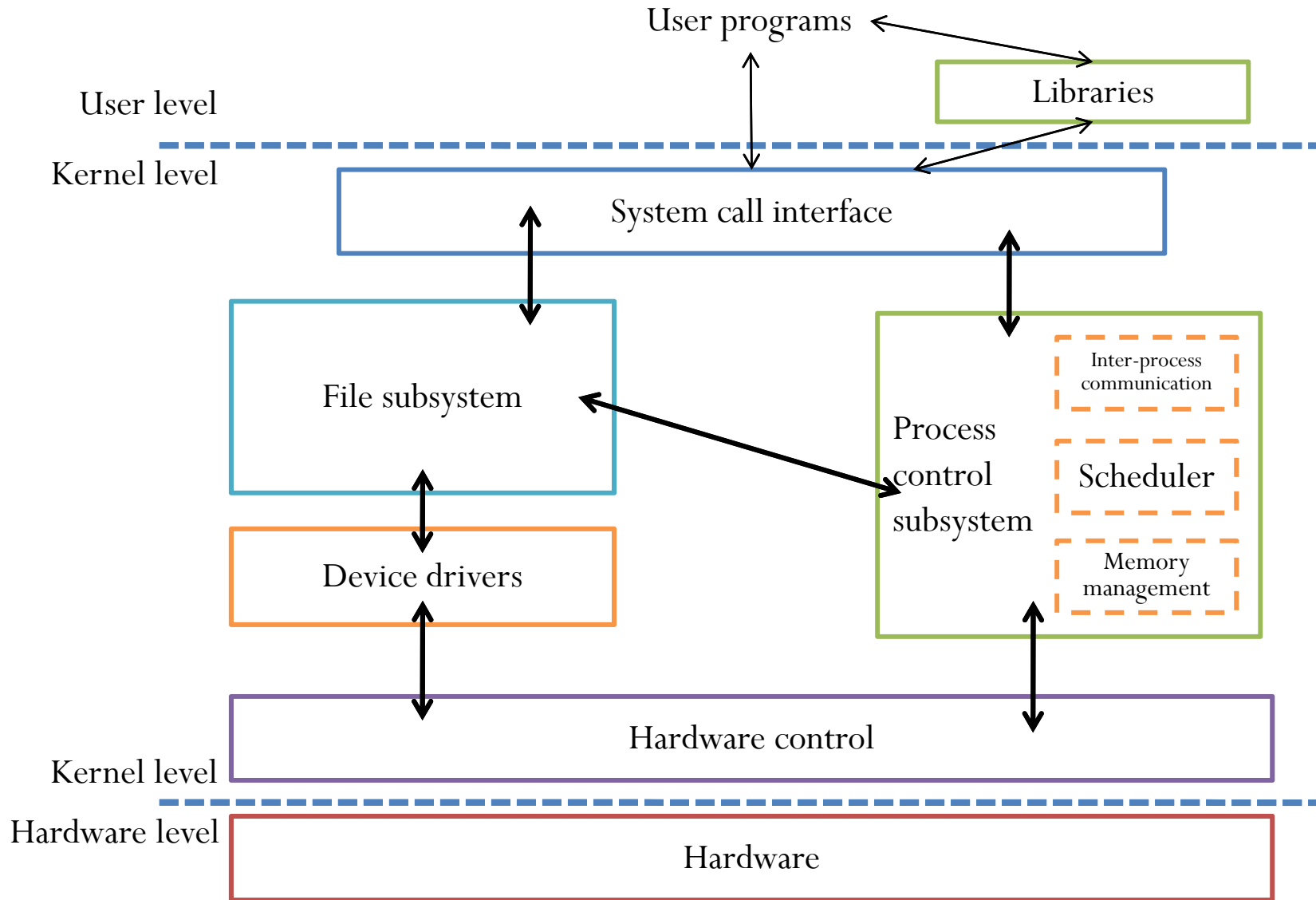
- Result

```
martini:~> gcc -o reader reader.c
martini:~> gcc -o writer writer.c
martini:~> reader &
[1] 22246
martini:~> writer &
[2] 22247
martini:~>
Reader, Print This Message!
[Enter]
[1]- Done      ./reader
[2]+ Done      ./writer
martini:~>
```

IPC Example (계속)



UNIX Kernel



Remaining Stuff

- Process Control System
 - Scheduler
 - Memory Management
- File System
 - Buffer Cache
- Miscellaneous
 - Error Handling
 - Time