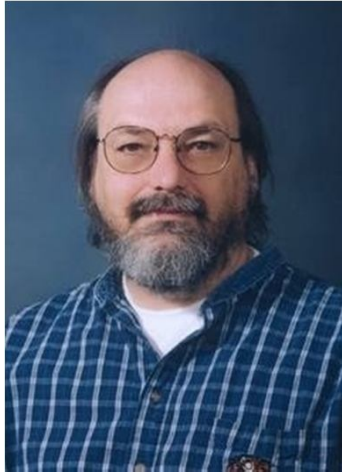




Introduction to UNIX



History of Unix



Kenneth Thompson

...

1966 -- Joins Bell Labs Computing Research Department, working on the Multics project

1969 -- Develops UNIX* operating system

1970 -- Writes B language, precursor to Dennis Ritchie's C language

1971 -- Moves UNIX from the PDP-7 to the PDP-11

1973 -- Rewrites UNIX in Dennis Ritchie's C language

...



Dennis M. Ritchie

...

1968 -- Joins the Bell Labs team working on Multics, a joint effort of Bell Labs, MIT and GE to develop a general computer operating system

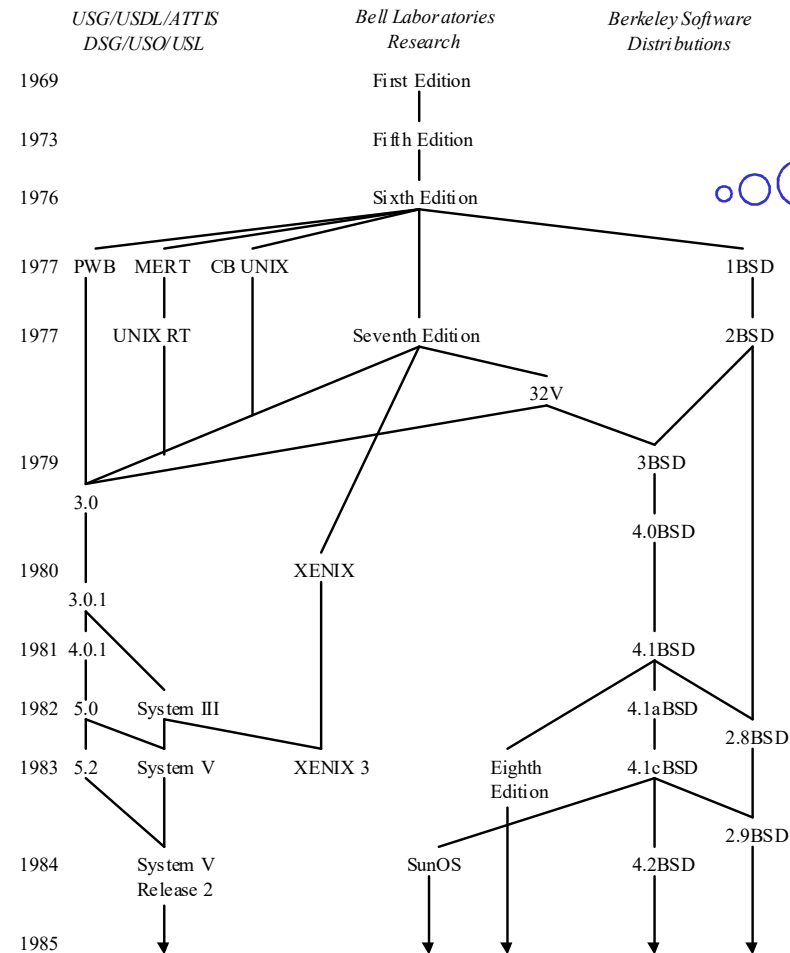
1972 -- Creates C language

...

History of Unix


- 📖 Bell Lab의 Ken Thompson이 PDP-7 위에서 개발(1969)
- 📖 Dennis Ritchie가 참여하여 대부분을 C언어로 작성.
- 📖 Sixth Edition 개발(1976)
 - 다른 기관이 사용할 수 있도록 배포한 최초의 버전
 - 이후 UNIX는 크게 AT&T 계열, BSD 계열로 분화해서 발전
- 📖 AT&T UNIX
 - System V, Release 3 개발(1987): STREAMS 도입
 - System V, Release 4 개발(1989): C/Korn shell, Job control, Symbolic link
- 📖 BSD UNIX
 - BSD 4 개발(1980): Job control, Reliable signal
 - BSD 4.4 개발(1993)

History of Unix



UNIX의 확산 계기
 1. 소스코드의 공개
 2. C언어로 작성
 (높은 이식성)

Figure 1.1 The UNIX system family tree, 1969-1985



There are too many variants of UNIX.
→ OS interface의 표준화 필요.

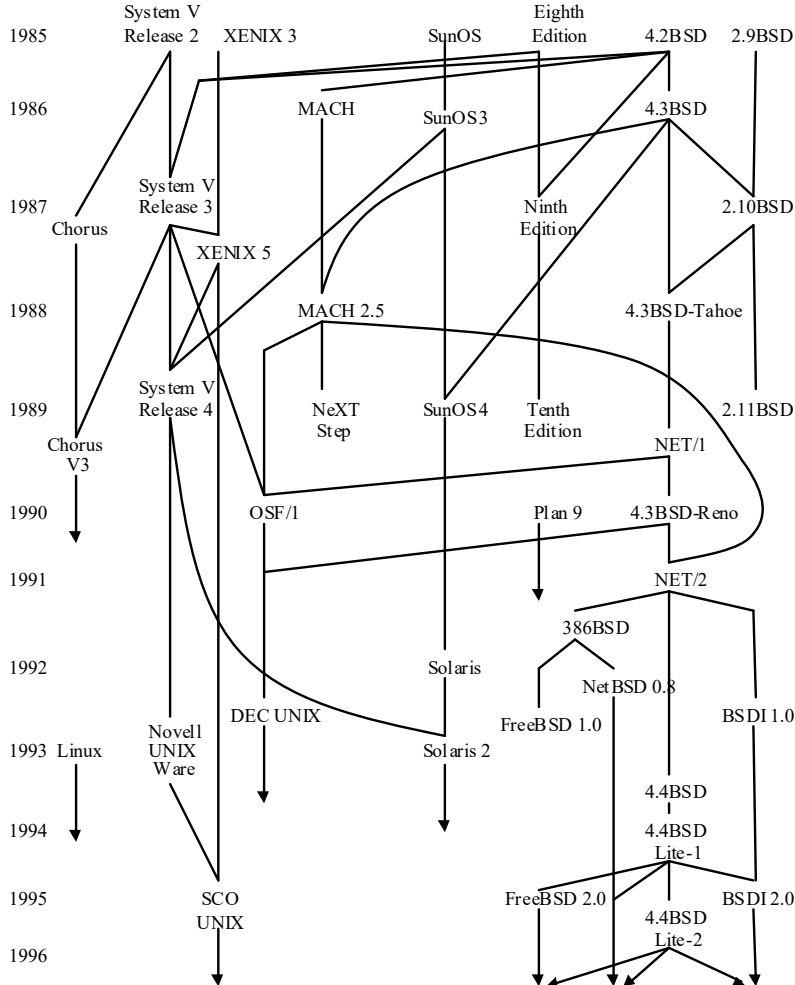


Figure 1.2 The UNIX system family tree, 1986-1996

UNIX의 표준화

ANSI C (1989)

- American National Standards Institute
- C 언어의 문법, 라이브러리와 헤더 파일의 표준을 제정
- 다양한 운영체제환경에서 C프로그램의 호환성을 높이기 위함

POSIX (1988)

- Portable Operating System Interface for Computer Environments
- 운영체제가 제공해야 하는 서비스를 정의
- 1003.1: 운영체제의 인터페이스 표준(POSIX.1 이라고도 함)
- 1003.2, 1003.7 등

Unix의 특징

What is Good with Unix?

- Open System
- Small is beautiful philosophy
 - file: just stream of bytes
 - Data, Device, Socket, Process, ... can be treated as a file.
- Portability
 - high-level language
 - client-server model, clustering, ...
- True Parallelism
 - Multitasking, Multiprocessor, ...

Unix의 특징

What is Wrong with Unix?

- Too many variants
 - dumping ground
- Not small and simple any more
 - uncontrolled growth
- Lack of GUI
 - not now (MIT's X)

What is Linux?

Linux

- PC 등 다양한 컴퓨터 환경에서 동작하는 UNIX-like OS
- 어느 상용 OS와도 견줄 수 있는 막강한 성능과 안정성
 - 인터넷 상의 많은 해커들의 참여로 커널이 개발됨.
 - 전세계의 수많은 사람들에 의해 테스트되고 개선, 발전됨.
- COPYLEFT(open source)
 - GNU General Public License (GPL)에 따라 공개
 - 프로그램의 실행파일 외에 소스코드 또한 공개.
 - 원하는 사람은 소스코드를 변경할 수 있음.
 - 단, 변경된 프로그램을 공개하고자 할 때는 반드시 소스코드도 함께 공개해야 함.

Linux의 역사

history

- 1991년 : 핀란드 대학원생 Linus Torvalds
 - Minix를 기반으로 version 0.01 개발
 - Paging, timer interrupt, device driver, file system, ...
- 1992년 : 리눅스 배포판 등장
 - 리눅스 배포판 = 리눅스 커널 + 응용 프로그램
- 1993년 : 커널 안정화
- 1994년 : version 1.0 발표
- 1996년 : version 2.0 발표
- 1999년 : version 2.2 발표
- 2001년 : version 2.4 발표
- 2003년 : version 2.6 발표
- 2012년 : version 3.0 발표
- 2016년 : version 4.0 발표



Linus Benedict Torvalds

Kernel 배포 사이트: <http://www.kernel.org>

Why Linux?

Why Linux?

- 강력한 기능(다음 페이지 참조)
- Unix based (POSIX 1003.1 standard 지원)
 - → Unix의 풍부한 S/W 사용 가능.
- open source code
 - → ‘experimenting’ with the system is possible.
- 풍부한 문서
 - FAQ, HOWTO 등

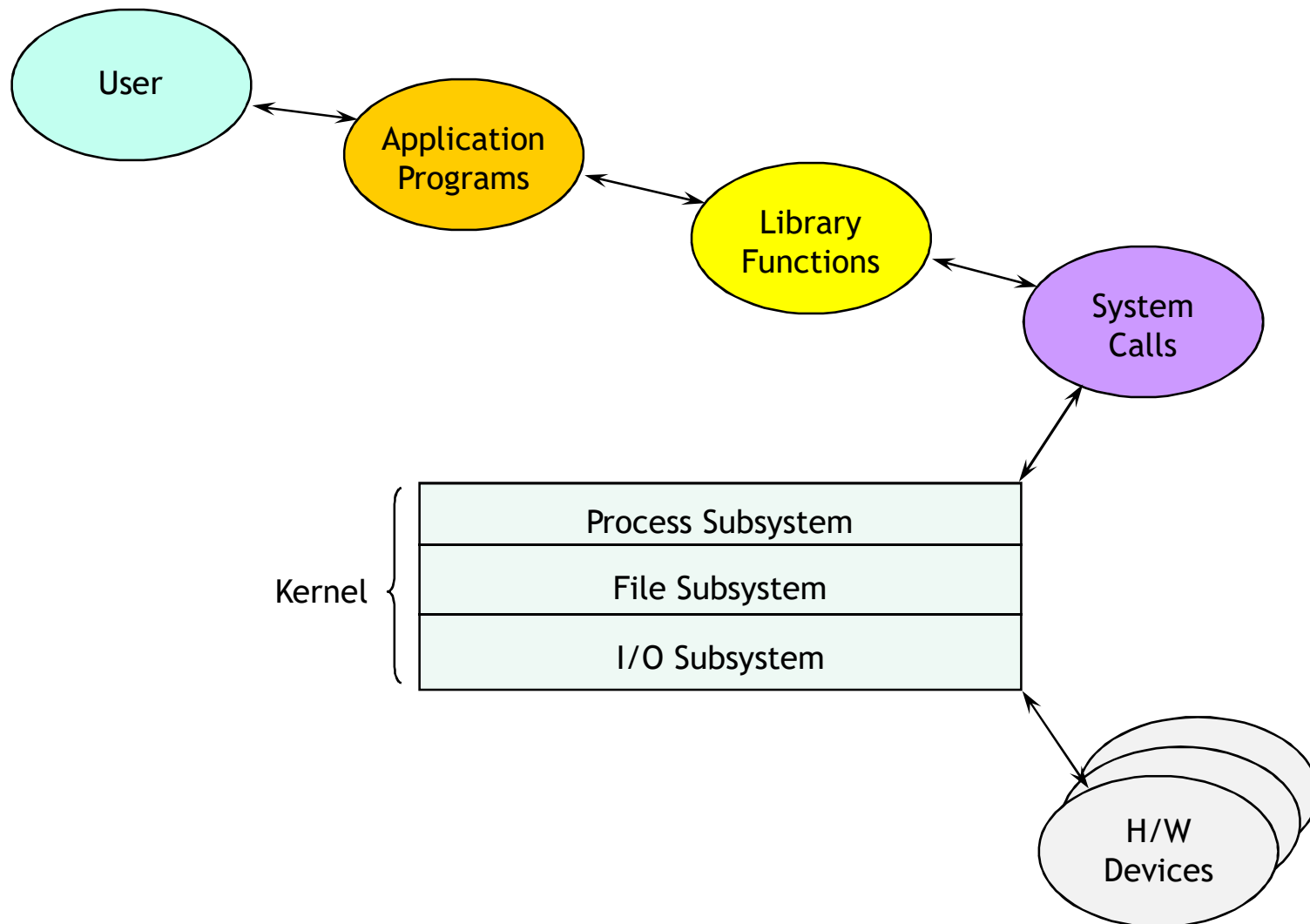
 → 실습 시 리눅스 사용

Why Linux?

리눅스의 기능상 장점

- 멀티 태스킹(Multi-Tasking)
- 다중 사용자 접근(Multi-User access)
- POSIX 1003.1 standard 지원
- 다양한 파일시스템 지원
 - EXT, JFS, ReiserFS, NTFS, ...
- 다양한 네트워크 프로토콜 지원
 - TCP/IP, SLIP, PPP, ...
- 다양한 아키텍처 지원
 - 80*86, SPARC, ARM, PPC,...
- 멀티 프로세서(Multi-processor) 지원
- ...

Unix/Linux의 구성



Unix/Linux의 구성

❏ Application program

- e.g. ls, mkdir, chmod, vi, sh

❏ Library

- 많은 library function은 결국 system call을 호출
- e.g. printf() → write()

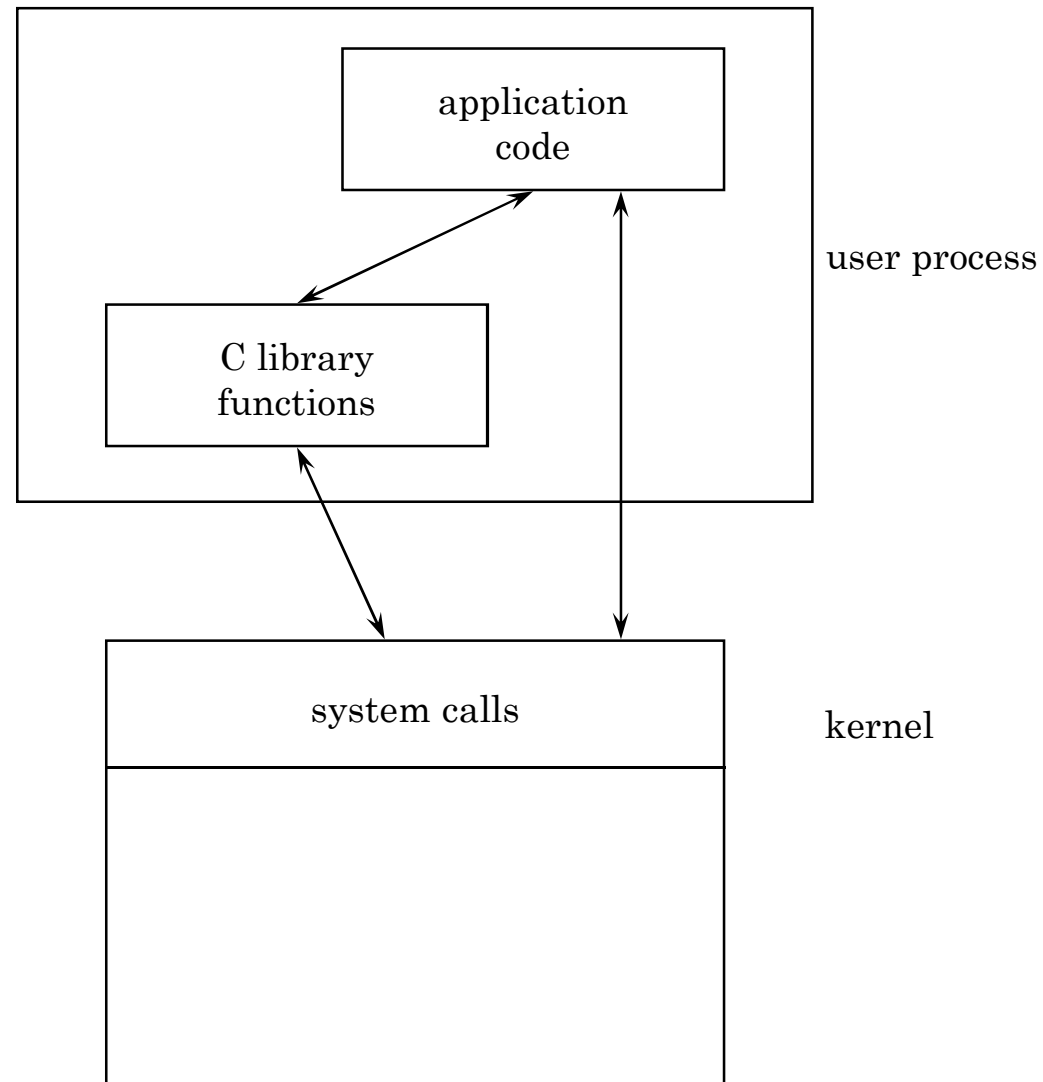
❏ System call

- Application과 operating system과의 interface.
- System call이 호출되면 kernel code가 수행됨.

❏ Kernel

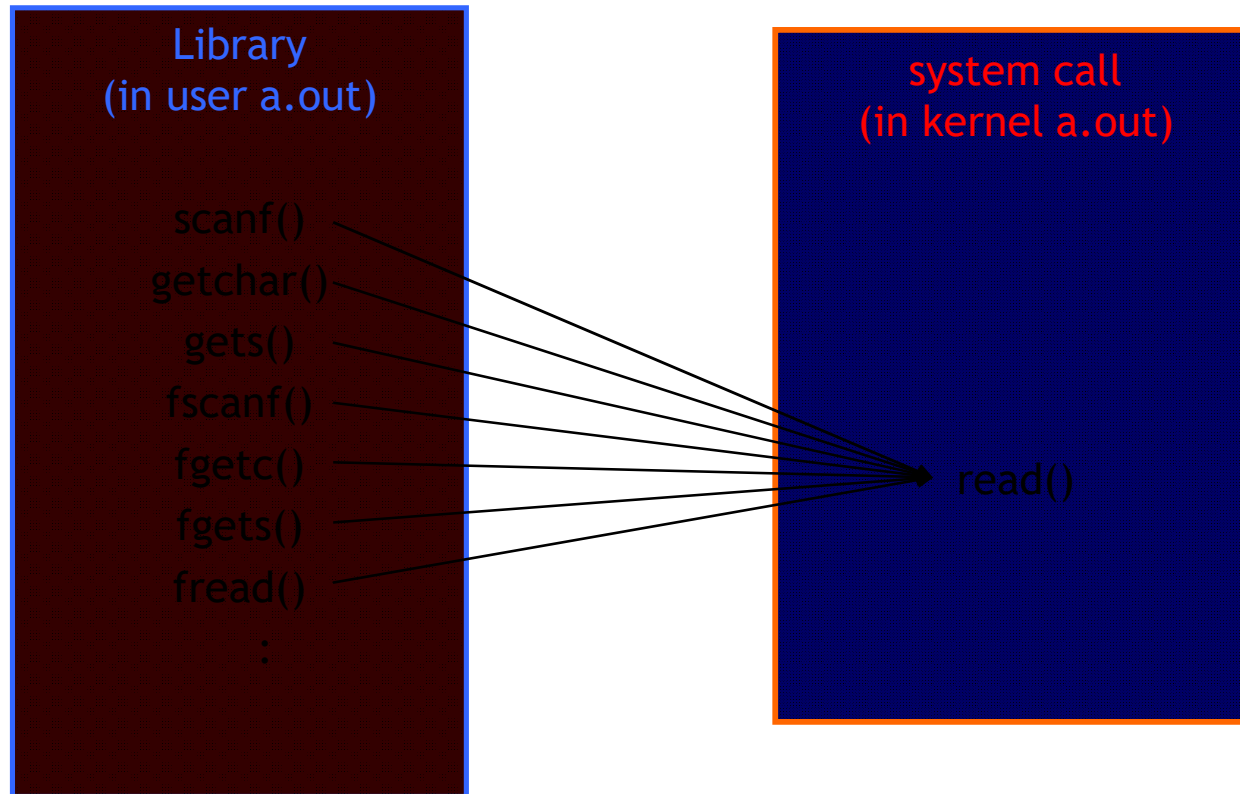
- system resource를 효율적으로 사용하도록 관리
- process/memory/file/IO management
- → 자세한 내용은 “운영체제” 과목에서 다룸.

Library vs. system call

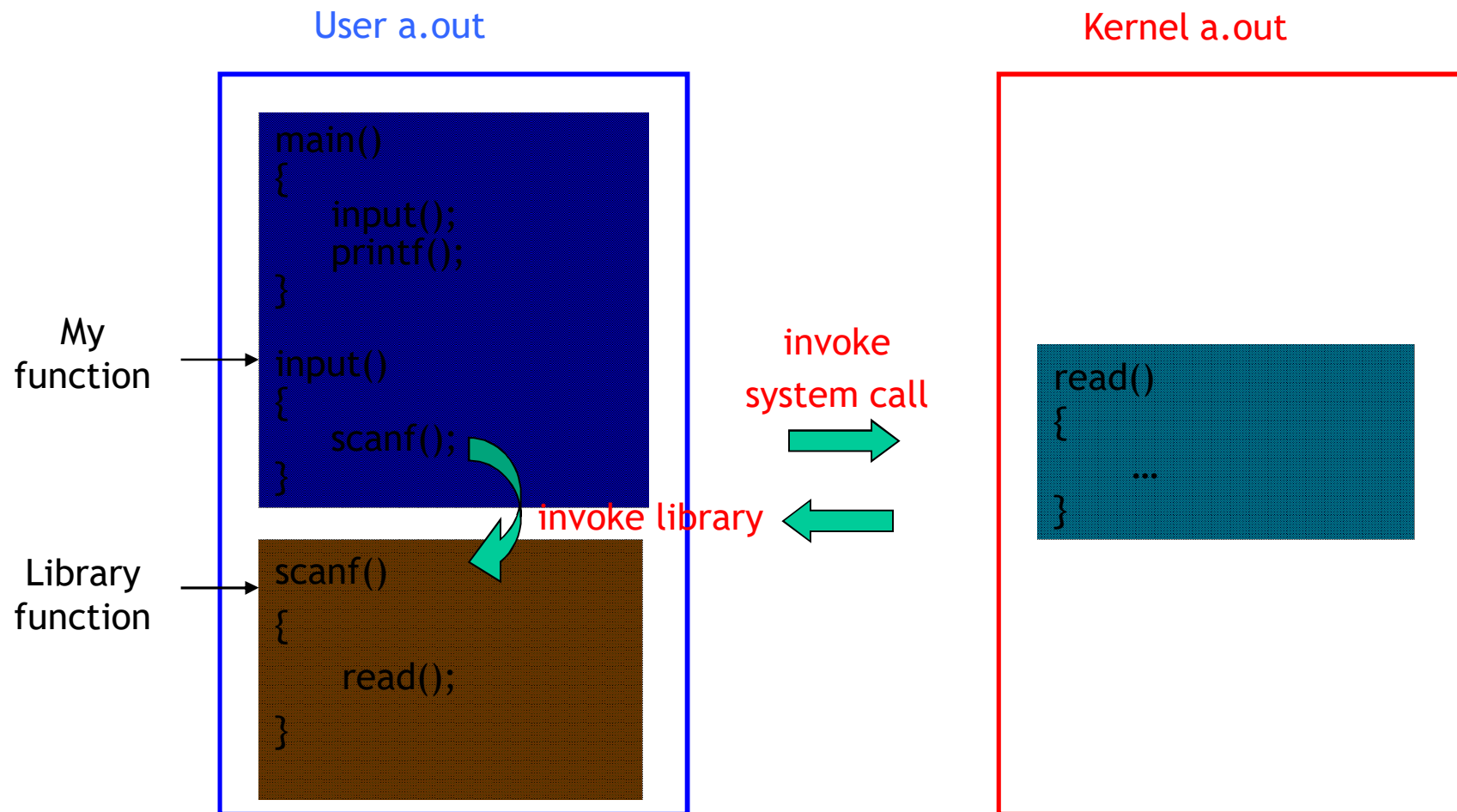


Library vs. system call

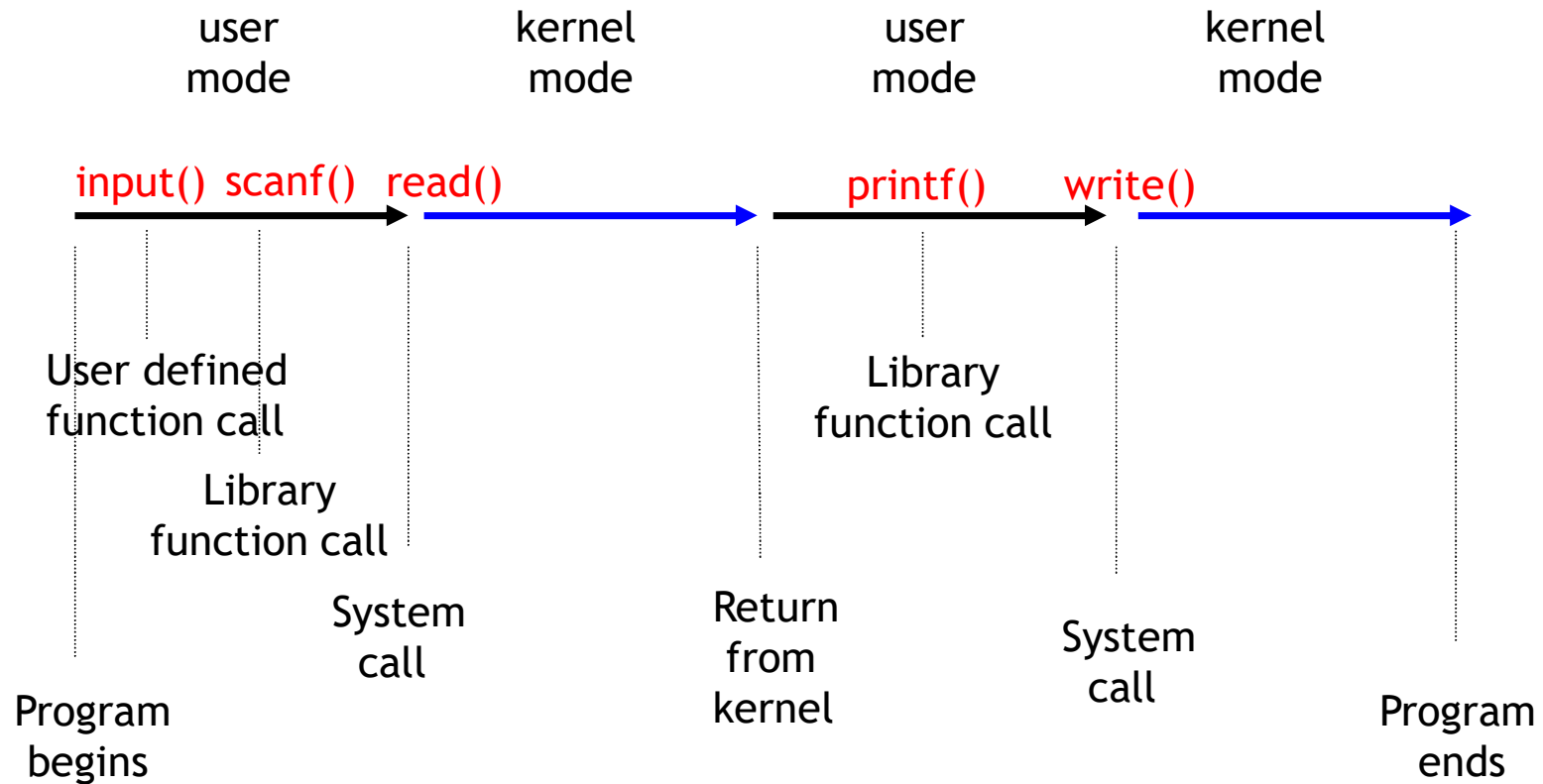
📖 Library와 system call의 예



Library vs. system call



Program execution



Kernel mode & User mode

kernel mode

- privileged mode
- no restriction is imposed on the kernel of the system
- may use all the instructions of the processor
- manipulate the whole of the memory
- talk directly to the peripheral controllers

Kernel mode & User mode

user mode

- normal execution mode for a process
- has no privileges
 - certain instructions are forbidden
 - only allowed to zones allocated to it
 - cannot interact with the physical machine
- process carries out operations in its own environment, without interfering with other processes
- process may be interrupted at any moment