

# 유닉스 시스템의 역사

## □ 유닉스 시스템의 역사

Unix 쪽에서 굉장히 유명한 사람.

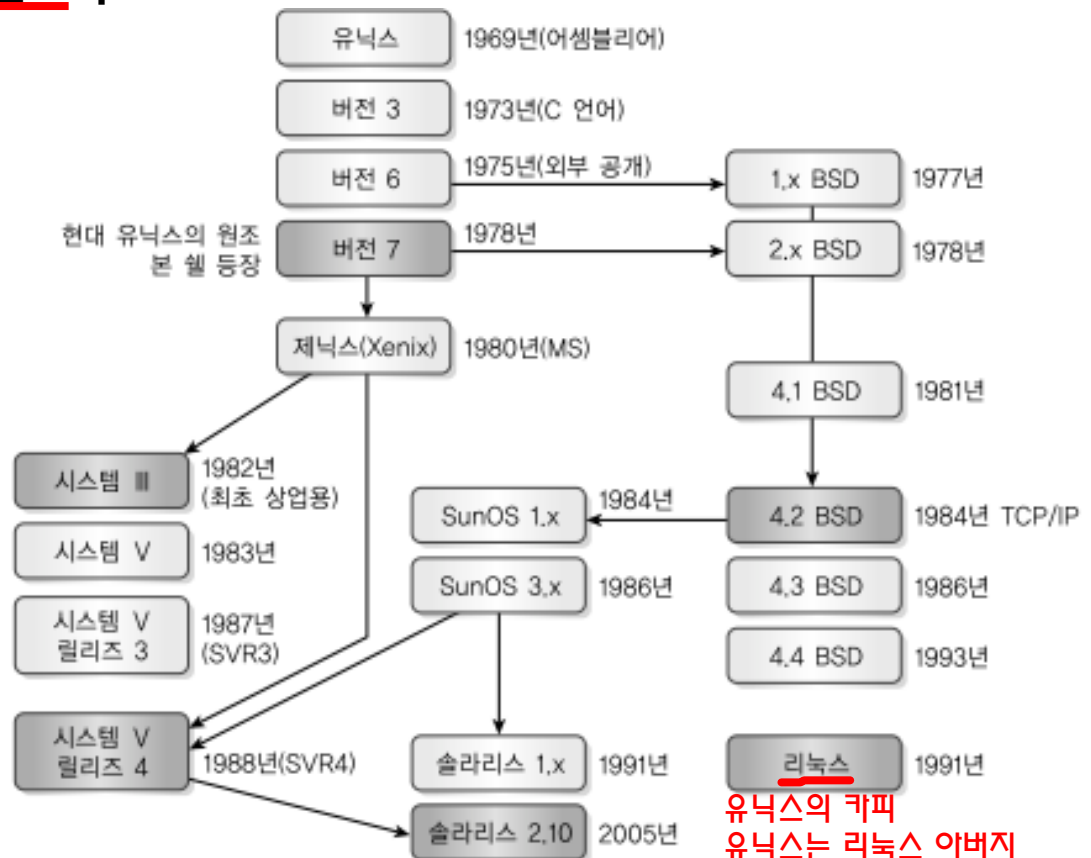
- 1969 AT&T산하의 벨연구소에서 켄 톰슨과 데니스 리치가 개발
- 1973 C언어를 이용하여 재개발 -> 고급 언어로 작성한 최초의 운영체제
- 그 후 상용유닉스(시스템V) 계열과 BSD 계열로 분리하여 각각 발전
- 1989 AT&T와 썬마이크로시스템즈가 <sup>세마포어</sup> 소켓의 개념이 처음 도입됨.

두 계열의 장점을 결합하여  
SVR4를 공동개발

- 이 유닉스가 현재 사용하는  
대부분의 유닉스의 기반임

멀티스 프로그래밍이 나왔지만 실패.

Unix 를 B언어로 개발했지만 C언어로 재개발.



## □ ANSI C 표준

- 미국 표준협회(ANSI)에서 표준화한 C 언어 명세 : ANS X3.159-1989
- ISO가 이를 받아들여 ISO/IEC 9899:1990으로 발표함([www.iso.org](http://www.iso.org))

## □ POSIX POSIX 기반의 시스템 콜을 사용할 예정

- 서로 다른 유닉스 시스템 사이에서 상호 이식이 가능한 응용프로그램을 개발하기 위한 표준으로 IEEE에서 제정
- POSIX.1(IEEE Std 1003.1) : C언어 응용 프로그래밍 인터페이스 표준
- POSIX.2(IEEE Std 1003.2) : 표준 셸과 유틸리티 프로그램 인터페이스 표준



## 준비사항

- [www.putty.org](http://www.putty.org)에서 putty.exe download
- IP address : 203.250.148.46
  - port:1074
  - s학번
  - passwd 수정



# 유닉스 시스템 프로그래밍이란

## □ 유닉스시스템 프로그래밍의 정의

- 유닉스에서 제공하는 시스템 호출을 사용해 프로그램을 작성하는 것을 의미

## □ 시스템 호출

- 유닉스 시스템이 제공하는 서비스를 이용해 프로그램을 작성할 수 있도록 제공되는 프로그래밍 인터페이스
- 기본적인 형태는 C 언어의 함수 형태로 제공

## □ 라이브러리 함수

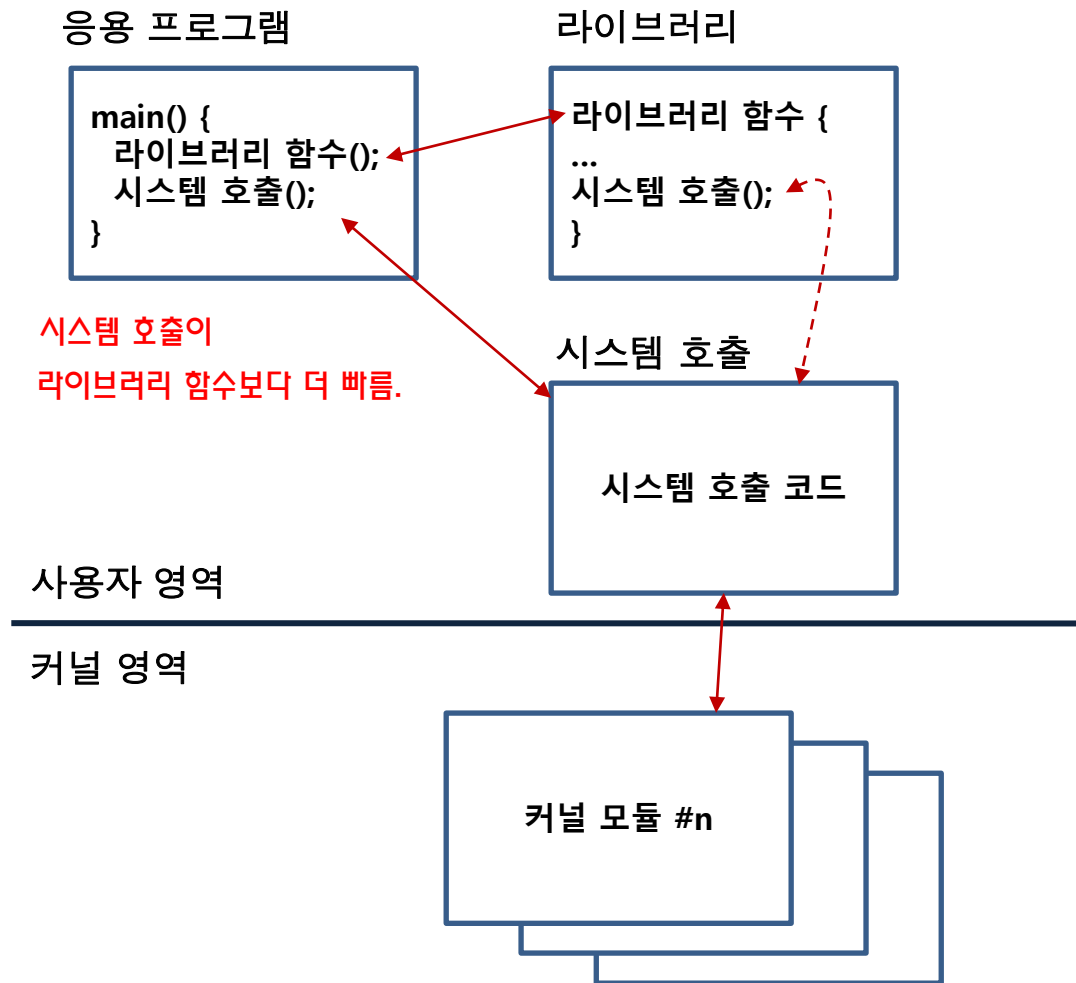
리턴값 = 시스템호출명(인자, ...);

- 라이브러리 : 미리 컴파일된 함수들을 묶어서 제공하는 특수한 형태의 파일
- 자주 사용하는 기능을 독립적으로 분리하여 구현해둌으로써 프로그램의 개발과 디버깅을 쉽게하고 컴파일을 좀 더 빠르게 할 수 있다
- /lib, /usr/lib에 위치하며 lib\*.a 또는 lib\*.so 형태로 제공



# 시스템 호출과 라이브러리 함수의 비교[1]

- 시스템 호출 : 커널의 해당 서비스 모듈을 직접 호출하여 작업하고 결과를 리턴
- 라이브러리 함수 : 일반적으로 커널 모듈을 직접 호출안함



## 시스템 호출과 라이브러리 함수의 비교[2]

### □ 시스템 호출 : man 페이지가 섹션 2에 속함

System Calls

open(2)

NAME

open, openat - open a file

SYNOPSIS

#include <sys/types.h>

### □ 라이브러리 함수 : man 페이지가 섹션 3에 속함

Standard C Library Functions

fopen(3C)

NAME

fopen - open a stream

SYNOPSIS

#include <stdio.h>

컴파일 할 때, 명령어 : gcc -o test1 test1.c

이 두개는 절대 순서가 바뀌어선 안됨!!

실행할 때, 명령어 : ./test1



# 시스템 호출과 라이브러리 함수의 비교[3]

## □ 시스템 호출의 오류 처리방법

- 성공하면 0을 리턴, 실패하면 -1을 리턴
- 전역변수 errno에 오류 코드 저장 : man 페이지에서 코드값 확인 가능

### [예제 1-1] 시스템 호출 오류 처리하기

ex1\_1.c

```
01 #include <unistd.h>
02 #include <stdio.h>
03
04 extern int errno;
05
06 int main(void) {
07     if (access("unix.txt", F_OK) == -1) {
08         printf("errno=%d\n", errno);
09     }
10
11     return 0;
12 }
```

```
# ex1_1.out
errno=2
```

```
# vi /usr/include/errno.h
.....
/*
 * Error codes
 */
#define EPERM    1      /* Not super-user */
#define ENOENT   2      /* No such file or directory */
.....
```



## 시스템 호출과 라이브러리 함수의 비교[4]

### □ 라이브러리 함수의 오류 처리방법

- 오류가 발생하면 NULL을 리턴, 함수의 리턴값이 int 형이면 -1 리턴
- errno 변수에 오류 코드 저장

[예제 1-2] 라이브러리 함수 오류 처리하기

ex1\_2.c

```
01 #include <stdlib.h>
02 #include <stdio.h>
03
04 extern int errno;
05
06 int main(void) {
07     FILE *fp;
08
09     if ((fp = fopen("unix.txt", "r")) == NULL) {
10         printf("errno=%d\n", errno);
11         exit(1);
12     }
13     fclose(fp);
14
15     return 0;
16 }
```

# ex1\_2.out  
errno=2

man fopen에서 확인



# 유닉스 기본 명령[1]

## □ 로그인/로그아웃

명령	기능	주요 옵션	예제
telnet	유닉스시스템에 접속	-	telnet hanb.co.kr
logout	유닉스시스템에서 접속해제	-	logout
exit		-	exit

## □ 프로세스 관련 명령

명령	기능	주요 옵션	예제
<u>ps</u>	현재 실행 중인 프로세스의 정보를 출력	-ef : 모든 프로세스에 대한 상세 정보 출력	ps ps -ef ps -ef   grep ftp
kill	프로세스 강제 종료	-9 : 강제 종료	kill 5000 kill -9 5001



# 유닉스 기본 명령[2]

## □ 파일/디렉토리 조작 명령

명령	기능	주요 옵션	예제
<u>pwd</u>	현재 <u>디렉토리 경로 출력</u>	-	pwd
<u>ls</u>	<u>디렉토리 내용 출력</u>	-a : 숨김파일출력 -l : 파일 상세정보 출력	ls -a /tmp ls -l
<u>cd</u>	현재 <u>디렉토리 변경</u>	.. : 부모 디렉토리로 감. .: 현재 디렉토리	cd /tmp <span style="color: red;">cd 0830</span> cd ~han01
<u>cp</u>	파일/디렉토리 <u>복사</u>	-r : 디렉토리 복사	<u>cp a.txt b.txt</u> <span style="color: red;">dif 명령어로</span> cp -r dir1 dir2 <span style="color: red;">같은지 확인 가능.</span>
<u>mv</u>	파일/디렉토리 <u>이름변경</u> 과 <u>이동</u>	-	<u>mv a.txt b.txt</u> <span style="color: red;">a.txt -&gt; b.txt</span> mv a.txt dir1 mv dir1 dir2
<u>rm</u>	파일/디렉토리 <u>삭제</u>	-r : 디렉토리 삭제 <span style="color: red;">-rf : 파일이 있는 디렉토리 강제 삭제</span>	rm a.txt rm -r dir1
<u>mkdir</u>	디렉토리 <u>생성</u>	-	mkdir dir1
<u>rmdir</u>	<u>빈 디렉토리 삭제</u>	-	mkdir dir2
<u>cat</u>	<u>파일 내용 출력</u>	-	cat a.txt
more	파일 내용을 쪽단위로 출력	-	more a.txt
chmod	파일 접근권한 변경	-	chmod 755 a.exe chmod go+x a.exe
grep	패턴 검색	-	grep abcd a.txt

# 유닉스 기본 명령[3]

파일 생성 : vi sample1.c

## □ vi 편집기 내부 명령

기능	명령	기능	명령
입력모드전환	<u>i,a,o,O</u> <small>입력하기 위해 'i'를 눌러야 함. 'o'를 입력하면 줄바꿈이 됨.</small>	<u>명령모드전환</u>	<Esc>
커서이동	j,k,h,l 또는 방향키	행이동	#G (50G, 143G 등) 또는 :행번호
한글자수정	r	여러글자수정	#s (5s, 7s 등)
단어수정	cw	명령취소	u, U
검색하여수정	:%s/aaa/bbb/g	복사	#yy (5yy, 10yy 등)
붙이기	p	커서이후삭제	D(shift-d)
<u>글자삭제</u>	<u>x</u> , #x(3x,5x 등) <small>입력 중 글자를 삭제하기 위해서는 'esc'를 누르고 'x'를 누름.</small>	<u>행삭제</u> (잘라내기)	누르기 전 꼭 'esc'를 눌러야 함. <u>dd</u> , #dd(3dd, 4dd 등)
<u>저장하고종료</u>	:wq! 또는 ZZ <small>w만 하면 저장만 함.</small>	저장않고종료	:q!
행 붙이기	J(shift-j)	화면다시표시	ctrl+l
행번호보이기	:set nu	행번호없애기	:set nonu



# 유닉스 기본 명령[4]

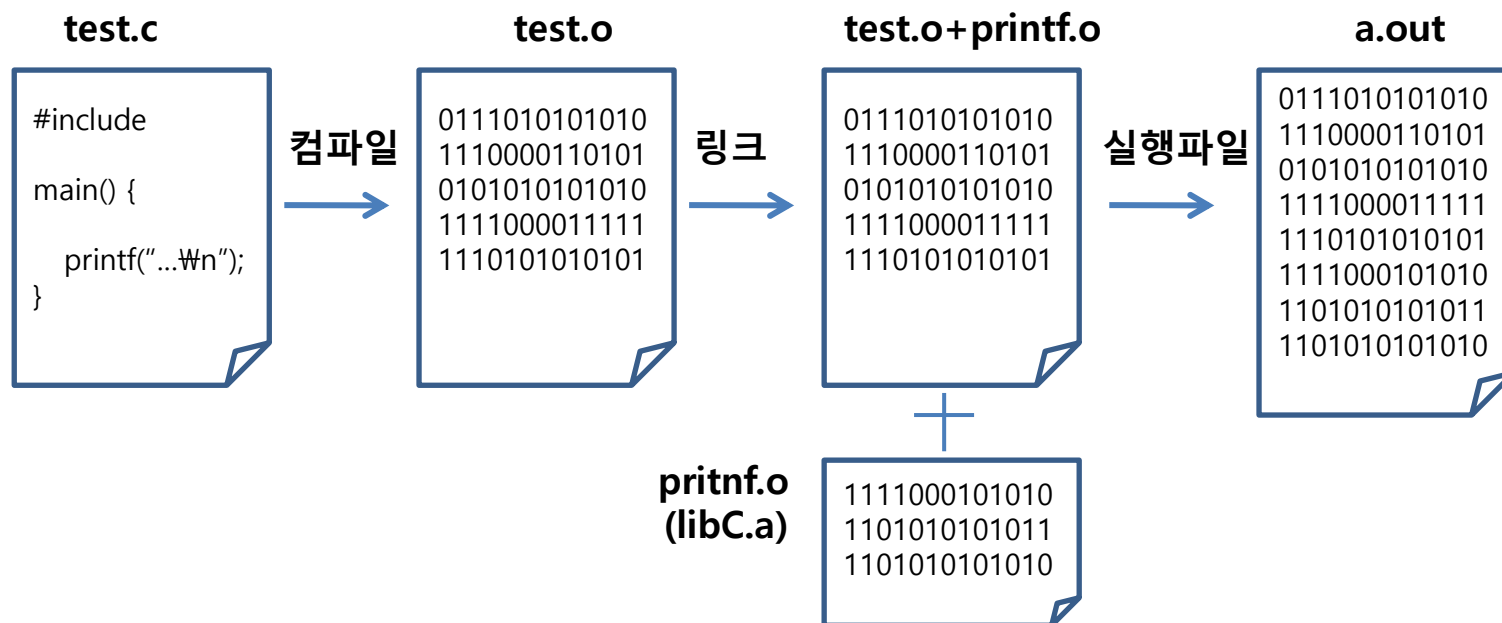
## □ 기타 명령

명령	기능	주요 옵션	예제
su	사용자 계정 변경	- : 변경할 사용자의 환경 초기화 파일 실행	su su - su - han02
tar	파일/디렉토리 묶기	-cvf : tar파일생성 -tvf : tar파일내용보기 -xvf : tar파일풀기	tar cvf a.tar * tar tvf a.tar tar xvf a.tar
whereis	파일 위치 검색	-	whereis ls
which		-	which telnet



## □ 컴파일이란


- 텍스트로 작성한 프로그램을 시스템이 이해할 수 있는 기계어로 변환하는 과정
- 보통 컴파일 과정과 라이브러리 링크 과정을 묶어서 수행하는 것을 의미




## 컴파일 환경[2]

### □ GNU C 컴파일러 : gcc

#### ■ C컴파일러 사용

  
# gcc test.c  
# ls  
a.out test.c

기본 실행파일명은  
a.out

  
# gcc -o test test.c  
# ls  
test test.c

실행파일명 지정은  
-o 옵션

