인덕대학교

운영체제 보안
- 프로세스 제한

컴퓨터소프트웨어학과

김병국 교수

학습목표

- 프로세스를 제어하기 위한 몇가지 명령어들을 사용할 수 있다.
- 프로세스의 우선순위를 변경할 수 있다.
- SSH 서버를 구축할 수 있다.
- 계정 별 기능을 제한할 수 있다.

목차

- 실습 환경 구축
- 프로세스 제어
- 프로세스 우선순위
- 계정 별 기능 제한

1. 실습 환경 구축 (1/2)

❑ 사용자 계정 추가

- 계정 이름: test
  - 계정의 프로세스를 제어하고 기능을 제한하기 위한 목적
  - 실행 명령어: adduser test

기존에 test 그룹이 이미 존재한다면?

- **그룹명의 중복으로 인해 계정 생성이 실패됨**
- **앞으로의 실습에 지장이 없으면 해당 그룹을 삭제할 필요가 있음**
  - 그룹 삭제 명령: delgroup <그룹명>

1. 실습 환경 구축 (2/2)

인덕대학교

❑코드 작성

- 프로세스의 상태 확인 및 제어를 위한 시험용 코드

```c
#include <stdio.h>
#include <unistd.h>

int main(void)
{
    for (int i = 0;; i++)
    {
        printf("\r");
        switch (i % 4)
        {
            case 0:
                printf("-");
                break;
            case 1:
                printf("\\");
                break;
            case 2:
                printf("|");
                break;
            case 3:
                printf("/");
                break;
        }
        fflush(stdout);
        if (i == 0x7FFFFFFF)
            i = 0;
    }
    return 0;
}
```

1

2

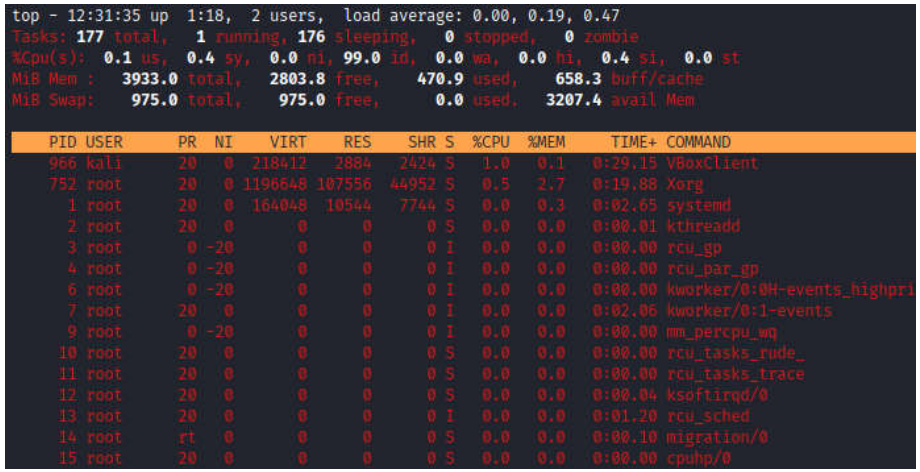
[파일명: loop.c]

실험 순서:
1. 컴파일: gcc loop.c –o loop
2. 파일 복사: sudo cp loop /home/test
3. 권한 허가: sudo chmod 755 /home/test/loop
4. 소유권 변경: sudo chown test.test /home/test/loop
5. test 계정으로 실험

2. 프로세스 제어 (1/3)

❑ 프로세스 모니터링

- 명령어: top
  - 프로세스의 상태정보를 정리된 화면으로 출력
  - 명령어 ps와 동일한 기능을 수행하나, 꾸준한 모니터링 및 사용자 요구에 맞는 형태로 계속 볼 수 있음

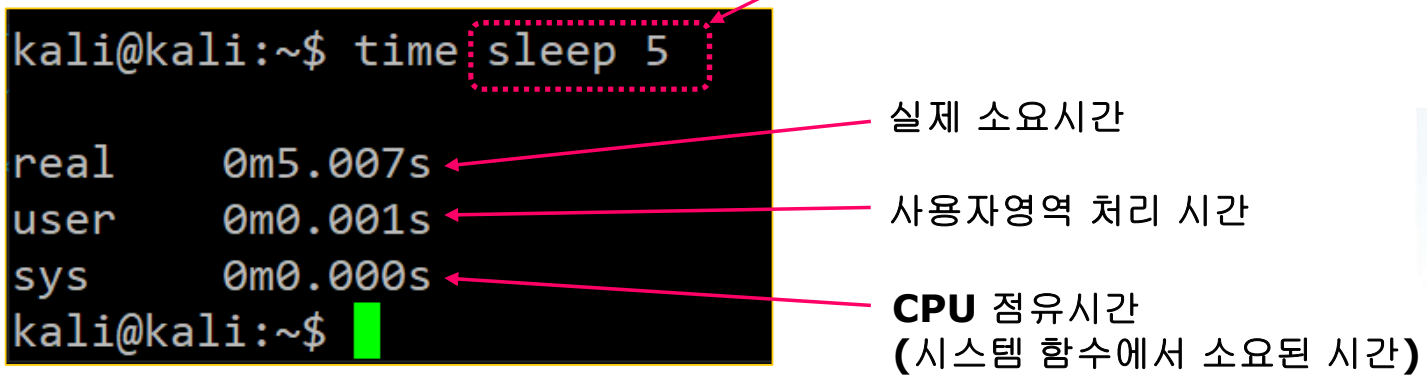


[top명령 실행 예]

- 주요 키:
  - <space bar> : 리스트 갱신
  - <home>, <end>, <pg-up>, <pg-dn> : 페이지 전환
  - u <username> : 지정한 사용자의 프로세스들만 표시 (단, 공백→ 모든 사용자)
  - z : 컬러모드 전환
  - V : 트리형태로 구성(v(소문자): 트리의 마지막 노드만 보임)
  - s <#> : 갱신주기 설정
  - k <#> : 프로세스에게 시그널 전송

❑ 프로세스 동작시간 파악

- 명령어: time [명령어]
  - 지정한 명령어에 대한 동작 시간을 측정



[time 명령 실행 예]

2. 프로세스 제어 (3/3)

❑ 프로세스 동작시간 설정

- 명령어: timeout {-옵션} [시간] [명령어]
  - 지정한 시간동안에만 프로세스가 수행
  - 시간 경과 후 강제 종료

```
[kali@kali ~]$
[kali@kali ~]$
[kali@kali ~]$timeout -k9 5s sleep 10
```

[timeout 명령 실행 예]

- 주요 옵션:
  - -s<#> : 지정한 타임아웃시간에 <#> 시그널을 전송
  - -k<#> : 타임아웃 이후에도 running 상태이면, <#>이후 KILL(9)신호를 보냄
- 시간 표현:
  - #s : 초단위
  - #m : 분단위
  - #h : 시간단위
  - #d : 일단위

4. 프로세스 우선순위 (1/5)

❑우선순위

- 프로세스는 고유의 우선순위(priority)를 가짐
- 값이 낮을 수록 높은 우선순위를 가짐

CPU 점유 기회가 더 높음

- 리눅스 운영체제 :
  - 새로운 프로세스를 생성시 부모의 우선순위를 적용
  - 값의 범위 : 0 ~ 39
  - 기본 값: 20

- 관련 명령: nice, renice

4. 프로세스 우선순위 (2/5)

❑ 우선순위 지정

- 명령어 : nice -n <오프셋> <명령어>
  - <명령어>가 실행될 때 20+< 오프셋>로 우선순위가 지정되어 실행됨
  - 오프셋이 -20보다 작으면 -20으로 처리됨
  - 오프셋이 19보다 크면, 19로 처리됨

❑ 우선 순위 변경

- 명령어 : renice <+/-오프셋> <프로세스ID>
  - 프로세스의 우선순위를 변경
  - 값의 범위: -20 ~ +19
  - **일반 사용자는 기존의 값에서 상향만 가능**
    - 설정파일을 통해 해당사항 예외가 가능
    - 설정파일: /etc/security/limits.conf

4. 프로세스 우선순위 (3/5)

❑ 실습(1/3)

- CPU 자원만 점유하는 의미 없는 코드 작성
- 파일명: load.c

```c
#include <stdio.h>
#include <time.h>

int main()
{
    do
    {
        time(NULL);
    } while (1);

    return 0;
}
```

[파일명: load.c]

- 컴파일: gcc load.c –o load
- 실행: ./load
- 강제 종료 키: [Ctrl + C]

4. 프로세스 우선순위 (4/5)

❑ 실습(2/3)

- 두 개의 프로세스에 대하여 우선순위 별 동작상태 확인
- 파일명: priority.c

```c
#include <stdio.h>
#include <string.h>
#include <time.h>
#include <unistd.h>

int ResultPrint(char *pBuffer);

int main()
{
    int nPid;
    char buffer[BUFSIZ];
```

1

```c
    memset(buffer, 0, BUFSIZ);

    nPid = fork();
    if (nPid > 0)
    {
        sprintf(buffer, "A (PID: %d)", getpid());
        ResultPrint(buffer);
    }
    else
    {
        sprintf(buffer, "\t\t\tB (PID: %d)", getpid());
        ResultPrint(buffer);
    }

    return 0;
}
```

2

[파일명: **priority.c (1/2)**]

4. 프로세스 우선순위 (5/5)

❑ 실습(3/3)

- 파일명: priority.c

```
30  int ResultPrint(char *pBuffer)
31  {
32      int nCount = 0;
33
34      for (int i = 0;; i++)
35      {
36          for (int j = 0;; j++)
37          {
38              time(NULL);
39              if (j >= 0x0FFFFFFF)
40                  break;
41          }
42          nCount++;
43          printf("%s (%d)\n", pBuffer, nCount);
44      }
45
46      return 0;
47  }
```

3

[파일명: **priority.c (2/2)**]

- 컴파일: gcc priority.c –o priority
- 준비 환경: 2개 이상 터미널(창) 실행(추천 4개)
  - 터미널 1: 코드 컴파일 및 실행용
  - 터미널 2: 로드 생성용
  - 터미널 3: 우선순위 확인용
  - 터미널 4: 우선순위 제어용
- 실행(터미널 1): ./priority

5. 사용자 제한용 환경구축 (1/2)

❑ 로그인 환경 구축

- SSH 서버
  - 대부분의 유닉스에서는 SSH(Secure Shell)를 지원함
  - 데비안 계열의 리눅스 배포판의 경우 APT(Advanced Package Tool)를 통해 관련 프로그램에 대한 설치 및 삭제 가능
  - 설치 확인
    - 명령: apt list *ssh*

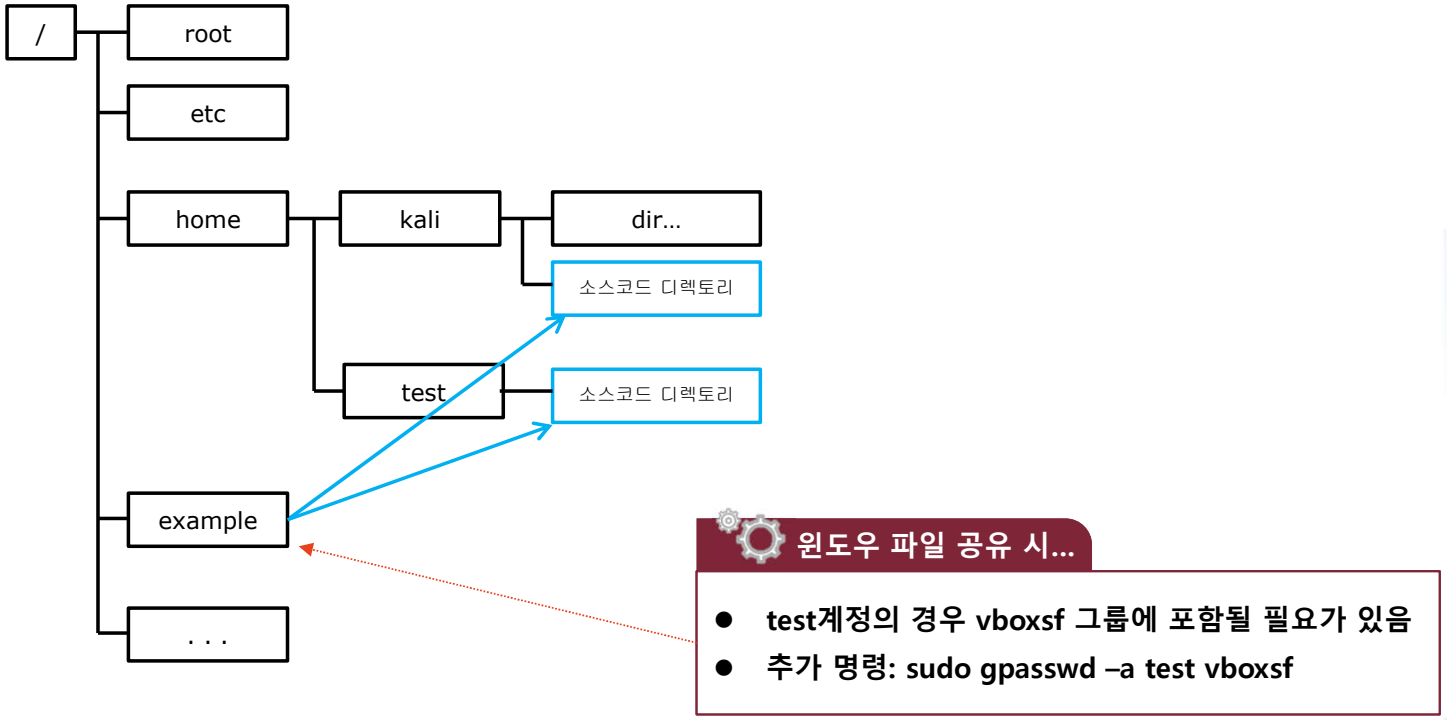
- SSH 서버 구동
  - 명령: sudo service ssh restart
  - 재부팅 후 자동 실행:

    sudo systemctl enable ssh

5. 사용자 제한용 환경구축 (2/2)

❑파일 접근 권한 및 링크

- 새로 추가된 계정(이름: test)의 접근을 허용하기 위한 선행작업 추천
- 작업중인 코드에 대한 test계정에서 쉬운 접근용 링크를 생성



6. 프로세스 제한 (1/10)

❑ 프로세스 제한

- 사용자의 명령들은 쉘(shell)을 통해 실행
- 쉘은 제한된 범주에서만 프로세스에게 자원을 할당
- 제한 설정 파일: /etc/security/limits.conf

❑ 제한 확인

- 명령어: ulimit {-옵션}
  - 제한 상태를 확인
  - 주요 옵션: -S (소프트), -H(하드), -a(모두 출력)

```
kali@kali:~$ ulimit -SHa
real-time non-blocking time  (microseconds, -R) unlimited
core file size              (blocks, -c) 0
data seg size               (kbytes, -d) unlimited
scheduling priority                 (-e) 0
file size                   (blocks, -f) unlimited
pending signals                     (-i) 15459
max locked memory           (kbytes, -l) 503420
max memory size             (kbytes, -m) unlimited
open files                          (-n) 1024
pipe size                (512 bytes, -p) 8
POSIX message queues         (bytes, -q) 819200
real-time priority                  (-r) 0
stack size                  (kbytes, -s) 8192
cpu time                   (seconds, -t) unlimited
max user processes                  (-u) 15459
virtual memory              (kbytes, -v) unlimited
file locks                          (-x) unlimited
kali@kali:~$
```

[ulimits 명령 실행 예]

6. 프로세스 제한 (2/10)

❑ 설정 파일내 구성

- 형태: <domain> <type> <item> <value>
- <domain> : 사용자 계정(사용자 명) 및 그룹(@그룹이름)을 지정
- <type> : soft와 hard로 구분(-를 사용 시 둘 다 사용함을 의미)
- <item> : 제한 기능 선택
- <value> : 제한 기능별 값

```
22 #<item> can be one of the following:
23 #        - core - limits the core file size (KB)
24 #        - data - max data size (KB)
25 #        - fsize - maximum filesize (KB)
26 #        - memlock - max locked-in-memory address space (KB)
27 #        - nofile - max number of open file descriptors
28 #        - rss - max resident set size (KB)
29 #        - stack - max stack size (KB)
30 #        - cpu - max CPU time (MIN)
31 #        - nproc - max number of processes
32 #        - as - address space limit (KB)
33 #        - maxlogins - max number of logins for this user
```

[item 예]

6. 프로세스 제한 (3/10)

❑최대 로그인 개수 제한

- 계정별 로그인 쉘의 개수를 제한

```
[test@kali ~]$w
 14:38:05 up  3:24,  4 users,  load average: 0.08, 0.08, 0.02
USER     TTY      FROM             LOGIN@   IDLE   JCPU   PCPU WHAT
kali     tty7     :0               11:13    3:24m 33.72s  0.19s xfce4-session
kali     pts/3    192.168.0.2      14:17   14:21   0.02s  0.02s -bash
test     pts/5    -                14:37   29.00s  0.04s  0.01s -bash
test     pts/6    -                14:38    5.00s  0.07s  0.00s w
[test@kali ~]$
```

```
kali@kali:~$ sudo login test
[sudo] password for kali:
Password:
Linux kali 5.10.0-kali3-amd64 #1 SMP Debian 5.10.13-1kali1 (2021-02-08) x86_64

The programs included with the Kali GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Kali GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
There were too many logins for 'test'.
Last login: Thu May 13 14:38:04 KST 2021 on pts/6

Permission denied
kali@kali:~$
```

[로그인 시도]

```
1  # /etc/security/limits.conf
2  #
3  #Each line describes a limit for a user in the form:
4  #
5  #<domain>        <type>  <item>  <value>
6  #
7  #Where:
8  #<domain> can be:
9  #        - a user name
10 #        - a group name, with @group syntax
11 #
12
13 test            soft    maxlogins       2
14
15 # End of file
```

[최대 로그인 개수 지정]

6. 프로세스 제한 (4/10)

❑CPU 점유시간 제한

- 특정 계정에 대한 CPU의 점유 시간을 제한
- Item의 값으로 cpu에 해당
- 단위 : 분(MIN)

```
1 # /etc/security/limits.conf
2 #
3 #Each line describes a limit for a user in the form:
4 #
5 #<domain>        <type>  <item>  <value>
6 #
7 #Where:
8 #<domain> can be:
9 #        - a user name
56
57 test            soft    cpu             1
58
59 # End of file
```

[CPU 점유시간 할당]

6. 프로세스 제한 (5/10)

❑ 프로세스 개수 제한

- 특정 계정에 대한 프로세스의 개수를 제한

```
53 #ftp                -       chroot          /ftp
54 #@student           -       maxlogins       4
55 #
56 test                soft    nproc           10
57
58 # End of file
```

[프로세스 개수 제한 설정 예]

```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>

int main(int argc, char *argv[])
{
    int nPid = 0;
    int nCount = 100;

    if (argc == 2)
        nCount = atoi(argv[1]);

    for (int i = 0; i < nCount; i++)  {
        nPid = fork();
        if (nPid == 0) {
            printf("[%02d] I'm a child process with %d.\n", i, getpid());
            sleep(100);
            break;
        }
    }

    return 0;
}
```

[파일명: **forkcount.c**]

순서:
1. 컴파일: gcc forkcount.c –o forkcount
2. 파일 복사: sudo cp forkcount /home/test
3. test 계정으로 실험

6. 프로세스 제한 (6/10)

❑파일의 크기 제한

①

```c
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>

int main(int argc, char *argv[])
{
    int nFd = -1;
    int nFileSize = 1024;
    int nOffset = 0;
```

```
#<domain>      <type>  <item>         <value>
#
test             soft    fsize           2000
```

[파일의 최대크기 제한 설정 예]

②

```c
    if (argc != 3)
    {
        printf("Usage: %s <filename> " \
        "<filesize(KB)>.\n", argv[0]);
        return -1;
    }

    nFileSize = atoi(argv[2]) * 1024;

    nFd = open(argv[1], O_WRONLY | O_CREAT, 0666);
    for (nOffset = 0; nOffset < nFileSize; nOffset++)
    {
        char ch = 'A';
        if (write(nFd, &ch, 1) < 0)
            break;
    }
    close(nFd);

    printf("We wrote %d bytes.\n", nOffset);

    return 0;
}
```

[파일명: **FileSize.c**]

- 컴파일: gcc FileSize.c –o FileSize
- 실행(test 계정): ./FileSize

6. 프로세스 제한 (7/10)

❑ 파일의 접속 개수 제한 (1/2)

1

```c
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>

#define MAX_FDS 1024
```

2

```c
int main(int argc, char *argv[])
{
    int p_nFd[MAX_FDS];
    int nFiles = 10;
    int nCount = 0;
    char p_Buffer[BUFSIZ];

    if (argc != 3)
    {
        printf("Usage: %s <filename> <File Count> ", argv[0]);
        return -1;
    }

    nFiles = atoi(argv[2]);
    if(nFiles>1024)
        nFiles = 1024;

    for (int i = 0; i < nFiles; i++)
    {
        p_nFd[i] = -1;
    }
```

[파일명: FileOpenCount.c (1/2)]

6. 프로세스 제한 (8/10)

❑ 파일의 접속 개수 제한 (2/2)

```
#<domain>      <type>  <item>         <value>
#
test             soft    nofile          20

#*               soft    core            0
```

[파일의 접근 개수를 제한 예]

- 컴파일: gcc FileOpenCount.c –o FileOpenCount
- 실행(test 계정): ./FileOpenCount
- 종료 후 반드시 : 생성파일들 삭제(rm 명령어 응용)

```
32  3
33      for (nCount = 0; nCount < nFiles; nCount++)
34      {
35          memset(p_Buffer, 0, BUFSIZ);
36          sprintf(p_Buffer, "%s_%d.txt", argv[1], nCount);
37          p_nFd[nCount] = open(p_Buffer, O_WRONLY | O_CREAT, 0666);
38          if (p_nFd[nCount]<0)
39              break;
40      }
41
42      printf("We Opened %d.\n", nCount);
43
44      for (int i = 0; i < nCount;i++)
45          close(p_nFd[i]);
46
47      return 0;
48  }
49
```

[파일명: **FileOpenCount.c (2/2)**]

6. 프로세스 제한 (9/10)

❑ 메모리 크기 제한

```
#<domain>      <type>  <item>         <value>
#
test            soft    data            10240
test            hard    data            10240
```

[메모리 크기 제한 예]

①

```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>

int main(int argc, char *argv[])
{
    char *p;
    int nDataSize = 1024;
    int nOffset = 0;
```

②

```c
    if (argc != 2)
    {
        printf("Usage: %s <datasize(KB)>.\n",
        argv[0]);
        return -1;
    }

    nDataSize = atoi(argv[1]);

    for (nOffset = 0; nOffset < nDataSize; nOffset++)
    {
        p = (char *)malloc(1024);
        if(p==NULL)
            break;
    }

    printf("We allocated %d kilo-bytes.\n", nOffset);

    getchar();

    return 0;
}
```

[파일명: **DataSize.c**]

- 컴파일: gcc DataSize.c –o DataSize
- 실행(test 계정): ./DataSize

6. 프로세스 제한 (10/10)

❑우선순위 지정

- 설정한 값이 기본값에서 가산 적용되어 프로세스가 동작됨

```
#<domain>      <type>  <item>          <value>
#

test            soft    priority        19
test            hard    priority        19
```

[우선순위 지정 예]

❑우선순위 제한

- 변경될 수 있는 우선순위의 범위를 제한함

```
#<domain>      <type>  <item>          <value>
#

test            hard    nice            -20
test            soft    nice            -20
```

[우선순위 제한 예]

수고하셨습니다.