UNIX 및 실습

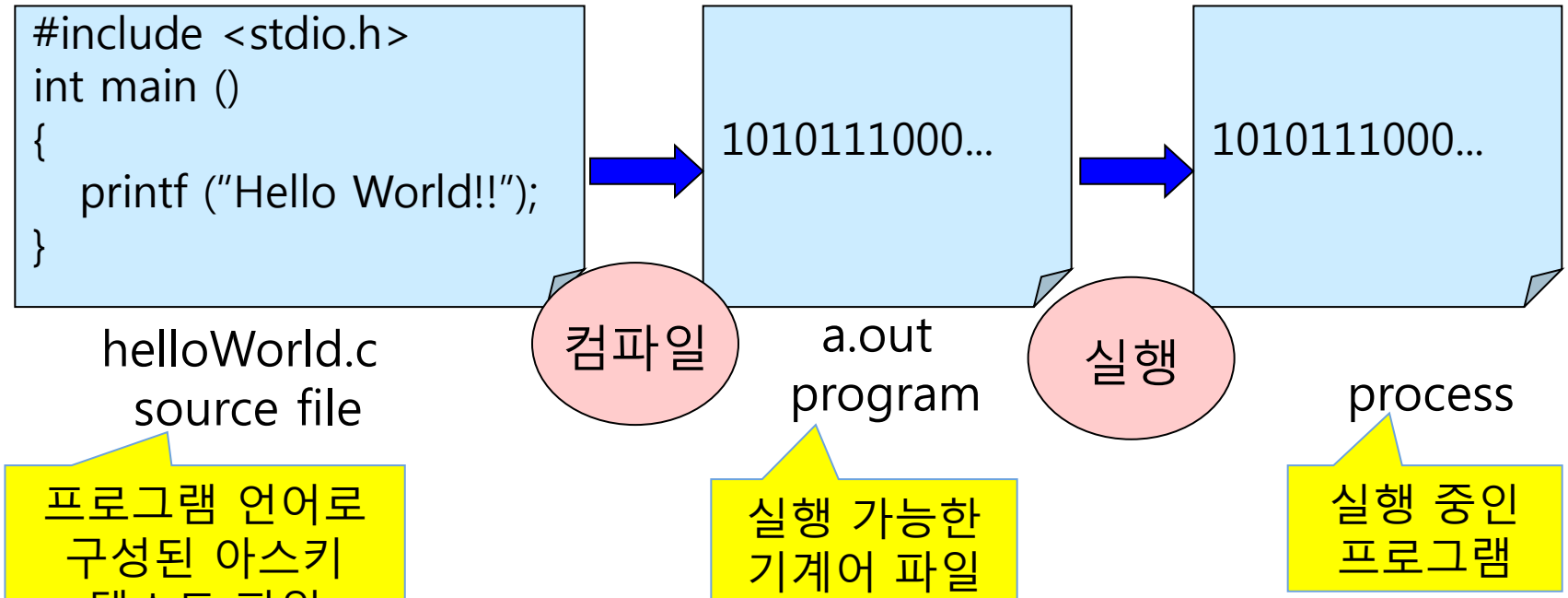
8장. 프로세스와 사용자 명령 익히기

학습목표

- 유닉스에서 프로세스가 무엇인지 그 개념을 이해한다.
- 프로세스와 관련된 명령의 사용 방법을 익힌다.
- 포그라운드 처리와 백그라운드 처리의 차이를 이해한다.
- 사용자 정보를 보는 명령의 사용 방법을 익힌다.

Section 01 프로세스란

- 프로세스(process)
  - 현재 시스템에서 실행 중인 프로그램
  - 프로세스는 고유 번호를 가진다.
    - Process ID : PID
    - 1번 프로세스 : init



프로세스의 종류

유닉스 프로세스의 종류

종류	설명
데몬 (daemon)	UNIX 커널에 의해 시작되는 프로세스로 서비스 제공을 위한 프로세스들이다.
부모 (parent)	자식 프로세스를 만드는 프로세스
자식 (child)	부모에 의해 생성된 프로세스로 실행이 끝나면 부모 프로세스로 돌아간다.
고아 (orphan)	자식프로세스가 종료하기 전에 부모가 종료된 프로세스. 고아프로세스는 1번 프로세스를 새로운 부모로 가진다.
좀비 (zombie)	부모프로세스가 종료처리를 하지 않은 프로세스 프로세스 테이블만 차지하고 있다.

Section 02 프로세스 관리

- 프로세스 목록 보기
  - ps
  - pgrep
- 프로세스 종료 시키기
  - kill
  - pkill
- 포그라운드(전위)와 백그라운드(후위) 작업제어
  - fg
  - bg
  - jobs

프로세스 목록보기 – ps [1/3]

ps [옵션]

- Process status
  - 프로세스 정보를 출력
  - PID, 터미널, CPU 시간, 명령어
- 옵션
  - -e : 시스템에 있는 모든 프로세스 목록 출력
  - -f : 프로세스에 대한 자세한 정보 출력
  - -u uid : 특정 사용자에 속한 모든 프로세스 출력

프로세스 목록보기 – ps [2/3]

사용법 [1]

ssh lily.mmu.ac.kr

```
[kgu@lily ~]$ ps
  PID TTY          TIME CMD
11607 pts/0    00:00:00 bash
11765 pts/0    00:00:00 ps
```

ssh lily.mmu.ac.kr

```
[kgu@lily ~]$ ps -e
  PID TTY          TIME CMD
    1 ?        00:00:17 systemd
    2 ?        00:00:01 kthreadd
    3 ?        00:00:01 ksoftirqd/0
    …
```

ssh lily.mmu.ac.kr

```
[kgu@lily ~]$ ps -f
UID         PID  PPID  C STIME TTY          TIME CMD
kgu       11607 11606  0 09:01 pts/0    00:00:00 -bash
kgu       11767 11607  0 09:18 pts/0    00:00:00 ps -f
```

프로세스 목록보기 – ps [3/3]

사용법 [2]

```
ssh lily.mmu.ac.kr
```

```
[kgu@lily ~]$ ps -ef |more
UID        PID  PPID  C STIME TTY          TIME CMD
root         1     0  0 Apr11 ?        00:00:17 /sbin/init
root         2     0  0 Apr11 ?        00:00:01 [kthreadd]
root         3     2  0 Apr11 ?        00:00:01 [ksoftirqd/0]
root         5     2  0 Apr11 ?        00:00:00 [kworker/0:0H]
root         7     2  0 Apr11 ?        00:00:00 [kworker/u:0H]
root         8     2  0 Apr11 ?        00:00:01 [migration/0]
```

구분	설명	구분	설명
UID	소유자의 사용자 ID	STIME	프로세스 시작시간
PID	프로세스 번호	TTY	터미널 번호(? : 데몬)
PPID	부모 프로세스 번호	TIME	CPU 사용시간
C	프로세스 우선순위	CMD	명령어 이름

특정프로세스 정보 검색하기 – pgrep [1/2]

pgrep ［옵션］ 패턴

- 프로세스 이름으로 찾아 정보를 출력
  - ' ps [옵션] | grep 패턴 ' 과 같은 기능
- 옵션
  - -x : 패턴과 정확히 일치하는 PID 출력
  - -n : 패턴을 포함하고 있는 가장 최근의 PID 출력
  - -U uid : 특정 사용자에 속한 PID 출력
  - -l : PID와 프로세스 이름 출력
  - -t term : 특정 터미널과 관련된 프로세스 출력
- 패턴
  - 찾으려는 정보

특정프로세스 정보 검색하기 – pgrep [2/2]

- 사용법

ssh lily.mmu.ac.kr

```
[kgu@lily ~]$ pgrep ssh
1096
10092
[kgu@lily ~]$ ps -e |grep ssh
 1096 ?        00:00:18 sshd
10092 ?        00:00:00 sshd
```

ssh lily.mmu.ac.kr

```
[kgu@lily ~]$ pgrep -n vi
10303
```

ssh lily.mmu.ac.kr

```
[kgu@lily ~]$ pgrep -l ssh
1096 sshd
10092 sshd
```

ssh lily.mmu.ac.kr

```
[kgu@lily ~]$ pgrep -lt pts/2
10098 bash
```

[실습하기] 프로세스 목록보기

- 실습하기

```
1)ps
2)ps -f
3)ps -e
4)ps -e | more
5)ps -ef | more
6)ps -ef | grep 로
  그인 id
7)ps -u 로그인id
```

```
8) ps -e | grep ssh
9)pgrep ssh
10)pgrep 로그인id
11)pgrep -l 로그인
   id
```

프로세스 종료시키기

- 프로세스의 종료
  - ps 명령으로 찾은 프로세스 중 불필요한 프로세스를 강제로 종료시킨다.
  - 프로세스를 종료시키면 그 자식 프로세스들도 같이 종료된다.
  - 프로세스를 종료시킬 때 PID나 프로세스 이름을 알아야 한다.
- 프로세스 종료 시키기
  - kill
  - pkill

시그널 [1/2]

- 시그널 (signal)
  - 프로세스에게 보내는 신호
  - 프로세스는 이 신호에 응답한다.
    - 신호 무시
    - 프로세스 종료 등
- kill, pkill 명령으로 신호를 보낸다.
- $ man signal 또는 man –s 5 signal로 자세한 정보를 찾아볼 수 있다.

시그널 [2/2]

- 시그널의 종류

시그널 번호	시그널 이름	기능	기본 응답
1	SIGHUP	• 터미널 연결이 끊어진 경우에 발생	종료
2	SIGINT	• 보통 Ctrl-C에 의해 발생	종료
9	SIGKILL	• 프로세스를 kill 시킨다. • 이 시그널은 무시할 수 없다.	종료
15	SIGTERM	• 프로세스를 종료시킨다. • 이 시그널은 무시할 수도 있다. • kill 명령이 보내는 기본 시그널	종료

프로세스 종료 – kill [1/2]

kill ［시그널］ pid

- 지정한 프로세스들에게 시그널을 보낸다.
  - 사용자가 소유한 프로세스만 종료시킬 수 있다.
  - root는 모든 프로세스를 종료시킬 수 있다.
- kill 명령은 디폴트로 15번(SIGTERM) 시그널을 보낸다 (soft kill).
- kill 명령을 사용하기 전에 대상 프로세스의 PID를 알고 있어야 한다 (ps, pgrep 명령 사용).
- 시그널
  - -9 : 강제종료

프로세스 종료 – kill [2/2]

- 사용법

```
ssh lily.mmu.ac.kr
[kgu@lily ~]$ sleep 1000 &
[1] 11787
[kgu@lily ~]$ kill 11787
```

soft kill
11787프로세스에게
SIGTERM 시그널을 보낸다.
이것이 바람직함

```
ssh lily.mmu.ac.kr
$ kill -9 11787
$
```

sure kill
11787 프로세스에게
SIGKILL 시그널을 보낸다.
SIGKILL을 받으면
프로세스가 즉시 종료된다.
어떤 시그널도 무시하는
프로세스를 종료시킬 때
유용하다.

프로세스 종료 – pkill [1/2]

pkill ［시그널］ 프로세스명

- pgrep 명령과 유사하게 패턴을 이용하여 프로세스를 찾아 해당 프로세스에게 시그널을 보낸다.
  - 사용자가 소유한 프로세스만 종료시킬 수 있다.
  - root는 모든 프로세스를 종료시킬 수 있다.
- kill 명령은 디폴트로 15번(SIGTERM) 시그널을 보낸다. (soft kill)
- kill 명령을 사용하기 전에 대상 프로세스명을 알고 있어야 한다. (ps, pgrep 명령 사용)
- 시그널
  - -9 : 강제종료

프로세스 종료 – pkill [2/2]

- 사용법

```
ssh lily.mmu.ac.kr
[kgu@lily ~]$ sleep 1000 &
[1] 11793
[kgu@lily ~]$
[kgu@lily ~]$ pkill sleep
[1]+  종료됨              sleep 1000
```

soft kill
sleep 명령을 수행하고 있는 프로세스에게 SIGTERM 시그널을 보낸다.

```
ssh lily.mmu.ac.kr
$ pkill -9 sleep
$
```

sure kill
SIGKILL 시그널을 보낸다.

[실습하기] 프로세스 종료

- 실습하기

```
1)ps
2)sleep 100 &
3)ps
4)kill  -9 PID(sleep)
5)vi /etc/hosts
* 다른 터미널에서
1)ps
2)kill -9  PID(vi)
```

```
1)ps
2)sleep 100 &
3)pkill  sleep
* 다른 터미널에서
1)vi /etc/hosts
2)pkill -9  vi
```

Section 03 포그라운드와 백그라운드 프로세스

- UNIX는 다중 작업을 지원하는 운영체제이다.
  - 동시에 여러 개의 작업을 수행할 수 있다.
- 포그라운드 처리(전위 처리)
  - 사용자가 명령을 입력한 후 결과가 출력될 때까지 기다려야 하는 경우
  - 보통의 명령처리 방법

```
ssh lily.mmu.ac.kr
$ find / -name passwd
```

- 백그라운드 처리(후위 처리)
  - 명령의 처리결과 출력과 관계없이 곧바로 프롬프트가 출력되어 다른 작업을 계속 할 수 있는 경우
  - 명령 실행시 마지막에 &를 붙임

```
ssh lily.mmu.ac.kr
$ find / -name passwd &
$
```

포그라운드와 백그라운드 작업제어

- 작업과 프로세스(job & process)
  - job은 쉘이 관리할 수 있는 프로세스이다.
  - 쉘은 job을 시작시키고 제어한다.
  - job은 프로세스이므로 각 job은 PID를 가지고 있고
  - 또한 쉘이 할당한 일련번호인 job ID도 가지고 있다.
  - 쉘은 동시에 여러 개의 job이 동작하도록 할 수 있다.
  - 포그라운드 작업 -> 포그라운드 프로세스
  - 백그라운드 작업 -> 백그라운드 프로세스
- 작업제어
  - 포그라운드 작업 -> 백그라운드 작업으로 전환
  - 백그라운드 작업 -> 포그라운드 작업으로 전환
  - 작업목록 보기
  - 작업 정지/종료/재동작

작업제어 명령 – jobs [1/2]

jobs [%작업번호]

- 백그라운드 작업을 모두 출력
- 특정 작업번호를 지정할 경우 해당 작업의 정보만 출력
- 작업번호
  - %번호 : 해당 번호의 작업 정보를 출력
  - %+ 또는 %% : 작업순서가 +인 작업 정보를 출력
  - %- : 작업 순서가 –인 작업 정보를 출력
- 사용법

```
ssh lily.mmu.ac.kr
```

```
$ jobs
[1] +  실행중 sleep 100 &
[2] -  실행중 sleep 200 &
$ jobs %1
[1] +  실행중 sleep 100 &
$
```

작업제어 명령 – jobs [2/2]

- jobs 명령 출력 항목

항목	출력예제	의 미
작업번호	[1]	작업번호로 백그라운드로 실행시킬 때마다 순차적으로 증가([1],[2],[3]...)
작업순서	+, -	작업순서를 표시 • + : 가장 최근에 접근한 작업 • - : + 작업보다 바로 전에 접근한 작업 • 공백 : 그 외의 작업
상태	실행중	작업의 상태를 표시 • 실행중(Running) : 현재 실행중 • 완료됨(Done) : 작업이 정상적으로 종료 • 종료됨(Terminated) : 작업이 비정상적으로 종료 • 정지(Stopped) : 작업이 잠시 중단됨.
명령	sleep 100&	실행중인 명령

작업제어 명령 - 작업전환 [1/2]

▶ 작업전환 및 종료 명령

명령	기능
bg [%작업번호]	현재 작업이나 특정 작업을 백그라운드로 전환시켜 실행
fg [%작업번호]	현재 작업이나 특정 작업을 포그라운드로 전환시켜 실행
ctrl+z	포그라운드 작업을 중지시키고, 백그라운드의 중지된 목록으로 보냄
stop %작업번호	백그라운드에서 수행중인 특정 작업을 중지
kill %n	특정 작업을 종료

작업제어 명령 - 작업전환 [2/2]

- 사용예

1

```
ssh lily.mmu.ac.kr
$ sleep 100
^Z
[1] + 정지(SIGTSTP) sleep 100
$ bg %1
[1]    sleep 100&
$
```

• 포그라운드 작업을 ctrl-z로 중지시키고 백그라운드로 전환시킨다.

2

```
ssh lily.mmu.ac.kr
$ jobs
[1] + 실행중    sleep 100&
$ fg
sleep 100
_
```

• jobs 명령으로 현재 작업을 확인한다.
• 1번 job : 실행중인 상태
• 포그라운드로 다시 전환하면 프로세스가 끝날 때까지 기다려야 한다.

[실습하기] 작업제어 명령

- 실습하기

1) vi /etc/hosts
   - ctrl+z
2) sleep 300&
3) jobs
4) kill %2
5) jobs
6) fg
7) vi 저장후 종료

8) sleep 150 &
9) sleep 200 &
10) jobs
11) fg %1
12) ^Z
13) jobs
14) kill %1
15) kill %2
16) jobs

작업제어 명령 - nohup

nohup 백그라운드명령

- 백그라운드 작업을 실행시킨 단말기가 종료되거나 사용자가 로그아웃하면 실행 중이던 백그라운드 작업은 함께 종료
  - 로그아웃한 다음에도 백그라운드 작업은 작업이 완료될 때까지 실행하도록 해야 할 때 nohup 명령 사용
- 명령의 실행결과와 오류메시지는 현재 디렉토리에 nohup.out파일로 자동적으로 저장
- 사용예

```
ssh lily.mmu.ac.kr
$ nohup find / -name passwd  &
[1]    16454
$ exit
```

nohup.out파일에 결과 저장.
다시 로그인해서 nohup.out파일 확인

Section 04 사용자 정보보기

- 로그인한 사용자 정보 보기
  - users, who, w
- 사용자 자신의 정보 보기
  - who am i, whoami, id

사용자명 출력하기 - users

users

- 현재 시스템에 로그인하고 있는 사용자명을 출력
- 사용예

```
ssh lily.mmu.ac.kr
```

```
$ users
root user1
$
```

사용자정보 출력하기 - who

who [옵션]

- 시스템을 사용하고 있는 사용자의 정보를 출력
- 옵션
  - -q : 사용자명만 출력한다.
  - -H : 출력항목의 제목도 함께 출력한다.
  - -b : 마지막으로 재부팅한 날짜와 시간을 출력한다.
- 사용예

```
ssh lily.mmu.ac.kr
[kgu@lily ~]$ who
(unknown) tty1         2013-04-11 17:58 (:0)
kgu      pts/0        2013-04-29 09:01 (220.68.173.204)
race0    pts/2        2013-04-28 23:23 (112.164.86.181)
```

```
ssh lily.mmu.ac.kr
[kgu@lily ~]$ who -q
(unknown) kgu race0 kgu race0 ww7737 kgu
# users=7
```

```
ssh lily.mmu.ac.kr
[kgu@lily ~]$ who -b
         system boot  2013-04-11 17:57
```

사용자 작업정보 출력하기- w

w [사용자명]

- 로그인한 사용자정보와 현재 하고 있는 작업정보를 출력
- 사용예

```
ssh lily.mmu.ac.kr
```

```
$ w
 09:13:36 up 17 days, 15:16,  7 users,  load average: 0.15, 0.13, 0.14
USER     TTY      FROM              LOGIN@   IDLE   JCPU   PCPU WHAT
kgu      pts/0    220.68.173.204   09:01    0.00s  0.15s  0.00s w
race0    pts/2    112.164.86.181   23:23    9:31m  0.17s  0.17s -bash
```

사용자 자신의 로그인 정보 – who am i

who am i

- who명령의 결과 중 자신에 대한 정보만 출력
- 사용예

```
ssh lily.mmu.ac.kr
```

```
[kgu@lily ~]$ who am i
kgu      pts/0        2013-04-29 09:01 (220.68.173.204)
```

자신의 로그인 사용자명 출력하기 – whoami

whoami

- 사용자의 로그인ID를 출력
  - BSD계열 명령
- 사용예

```
ssh lily.mmu.ac.kr
```

```
[kgu@lily ~]$ whoami
kgu
```

현재 사용자명과 그룹정보 출력하기 – id

id [옵션]

- 사용자의 로그인ID와 그룹 정보를 출력
- 옵션
  - -a : 기본 그룹 외에 2차 그룹 정보도 출력
- 사용예

```
ssh lily.mmu.ac.kr
```

```
[kgu@lily ~]$ id race0
uid=10044(race0) gid=10000(students) groups=10000(students)
[kgu@lily ~]$ id -a
uid=1000(kgu) gid=1000(prof) groups=1000(prof),3(sys),4(adm),10043(svn)
context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
```

[실습하기] 사용자정보 보기

- 실습하기

```
1)who
2)who -m
3)who -q
4)who -H
5)w
6)who am i
7)whoami
8)id
9)id -a
```