9강. 프로세스 관리 명령(2) & 쉘

- 포어그라운드(foreground) 프로세스
- 쉘 프롬프트에서 명령을 입력하고 <Enter>하면 포어그라운드 모드로 실행됨
- 포어그라운드 프로세스는 키보드 입력 받기와 화면 출력이 가능함
- 포어그라운드로 실행 중인 프로세스의 강제 중지는 <Ctrl>+<C>
- 포어그라운드로 실행 중에 <Ctrl>+<Z>하면 백그라운드로 전환되며 Stopped 상태가 됨
- 'fg *jobId*' 또는 'bg *jobId*' 명령을 수행하면 Stopped 백그라운드 프로세스가 실행상태가 될 수 있음

- 백그라운드(background) 프로세스
- 쉘 프롬프트에서 명령 뒤에 '&'를 적고 <Enter>하면 백그라운드로 실행됨
- 백그라운드로 프로세스를 실행시키면 쉘은 즉시 명령 대기 상태가 됨
- 백그라운드 프로세스는 키보드 입력을 받을 수 없음
- 입력 없이 장시간 실행되어야 하는 경우 주로 사용
- 포어그라운드로 전환시키려면 쉘에서 'fg jobId' 명령을 실행
- jobs 명령으로 백그라운드 프로세스의 상태를 점검할 수 있음

- 프로세스 모니터링 명령 - ps
- 실행 중인 프로세스에 관한 정보를 보여줌
- ps 명령의 사용 방법과 옵션

ps [옵션]

◦ l : 세부적인 정보 (교재 참고)
◦ a : 다른 사용자의 프로세스 포함
◦ u : 상세한 사용(자)정보 포함
◦ x : 터미널(tty)이 할당되지 않은 프로세스 포함
◦ -p *pid* : PID가 *pid*인 프로세스만 해당
◦ -u *uid* : UID(또는 사용자계정)가 *uid*인 사용자의 프로세스만 해당

- 프로세스 모니터링 명령 - top
- 프로세스 모니터링과 프로세스 관리를 위한 대화식 툴
- 시스템에서 실행되고 있는 프로세스들의 실시간 정보(3초 간격)를 보여줌
- 전체적 CPU 사용, 메모리 사용, 시스템 부하 등과 개별 프로세스의 실행 현황 등
- 윈도우의 작업 관리자와 비슷함
- top 명령의 사용 방법과 옵션

top [옵션]

◦ top의 대화식 서브 명령
◦ h : 도움 화면
◦ d : 화면 갱신 주기 조정

◦ u : 특정 사용자의 프로세스로 한정
◦ r : 우선 순위 조정(renice)
◦ k : 특정 프로세스(PID)를 kill

● 프로세스 관리 명령 - kil/killall
• kill 명령은 프로세스에 시그널(기본 SIGTERM)을 보내는 명령
• kill 명령은 대상 프로세스를 PID로 지정함. killall 명령은 명령 이름으로 지정함
• kill 명령의 사용 방법과 옵션
kill [옵션] *pid*
◦ -s signal 또는 -signal : 프로세스에 signal 시그널을 보냄 (SIGHUP, SIGKILL 등의 이름 또는 시그널의 번호)
◦ -l : 지정 가능한 시그널 이름을 보여줌
• killall 명령의 사용 방법
killall [옵션] 명령이름 : 여러 프로세스를 kill시킬 수 있으므로 주의해야 함

● 프로세스 관리 명령 - exec
• exec 명령을 호출하는 프로세스를 새로운 프로그램을 수행하도록 변경
-새로운 프로세스를 만들지 않고 새로운 명령을 수행하는 프로세스로 바꾸는 것
-쉘에서 exec 명령을 실행하면 명령의 실행이 종료된 후 쉘이 종료됨
• exec 명령의 사용 방법
exec [[옵션] 명령 [인수]]
◦ 예: exec top

● 프로세스 관리 명령 - nice (관련 명령 renice)
• 프로세스를 실행할 때 우선순위 값을 설정
• nice를 통해 조정될 수 있는 범위
◦ -20(가장 높은 우선권)에서 +19(가장 낮은 우선권)까지, 기본은 0
◦ 일반 사용자는 자신의 프로세스를 0 이상으로만 지정 가능
• nice 명령의 사용 방법과 옵션
nice [옵션] [명령 [인수]]
◦ -n adjustment 또는 -adjustment : 조정 수치를 adjustment 로 지정
◦ 실행 예: nice -n +5 top 또는 nice -+5 top
• renice 명령은 이미 실행 중인 프로세스의 우선순위 값을 설정
◦ 실행 예: renice +5 -p PID

● 프로세스 관리 명령 - nohup
• 터미널을 빠져나가도(예: 로그아웃) 실행 중인 프로그램을 종료되지 않고 계속 수행되게 함
• nohup으로 명령을 실행하는 경우 명령 행의 끝에 '&'를 붙여 백그라운드로 실행하고 로그아웃 함
• nohup 명령의 사용 방법
◦ nohup 명령 [인수]

◦ 실행 예: nohup find / -name core -print > corefiles.out &

- 프로세스 관리 명령 - cron
- 특정 시간에 특정 작업을 자동으로 수행하도록 지정 가능
- crond에 의해 수행될 작업 목록을 작성해야 함
- 일반적으로 수행될 작업을 스크립트로 작성해 두고, 스크립트를 수행하도록 함
- cron을 이용한 자동 스케줄과 /etc/crontab 파일 : 시스템 수준에서 주기적으로 수행될 작업을 설정
- crontab 명령 : 개별 사용자는 crontab 명령을 이용하여 주기적으로 수행될 작업을 설정할 수 있음

- crontab 파일

◦ 작성 예

```
02 4 * * * root run-parts /etc/cron.daily
```

- crontab 파일에서 필드가 표현하는 값(왼쪽부터)

◦ 분 : 0-59, *
◦ 시간 : 0-23, *
◦ 일 : 1-31, *
◦ 월 : 1-12, *
◦ 요일 : 0-6(0=일요일)
◦ run-parts는 해당 디렉터리에 있는 명령을 수행하는 명령

- crontab 명령의 사용 방법과 옵션

crontab [-u 사용자ID] [옵션]

◦ -l : crontab 파일 내용 보기
◦ -e : crontab 파일 편집
◦ -r : crontab 파일 삭제
◦ 예: '0 4 1-10 * * 명령1'은 매월 1일부터 10까지 매일 4시 0분에 '명령1'을 수행
◦ 예: '* * * * * date >> ~/date.txt'는 1분 간격으로 시간을 홈디렉터리의 date.txt에 추가

- crontab 명령 사용 예
- crontab -e (crontab 파일 생성)
- '* * * * * date >> ~/date.txt'을 작성하고 저장 후 종료
- crontab -l (crontab 파일 확인)
- cat ~/date.txt (홈디렉터리의 date.txt 파일을 확인)
  Wed Sep 28 14:00:02 KST 2011
  Wed Sep 28 14:01:01 KST 2011
- crontab -r (crontab 파일 삭제)

- 데몬
- 백그라운드로 실행되며 커널 요청 시 작동하는 프로세스

- 주로 시스템의 서비스를 제공

● 주요 데몬의 종류와 기능
- crond : 시간에 따라 등록된 명령을 정기적으로 실행
- dhcp : 동적인 IP 주소와 네트워크 정보를 지원
- httpd : 웹 서버
- lpd : 프린트 작업 처리
- sendmail : 메일 전송 서버

● 쉘 개요
- 쉘은 리눅스의 대화형 사용자 인터페이스
- 쉘은 명령어 해석기로 사용자의 명령을 해석하여 운영체제에 전달하고 명령 수행 결과를 보여줌
- 쉘은 스크립트를 해석하여 실행시킬 수 있음
- 쉘 스크립트는 프로그램과 유사함

● 쉘의 종류
- Bourne Shell (sh)

-UNIX 시스템의 표준 구성 요소
-Bourne Shell 프로그램의 실행명령어 : sh
-'.profile' 파일로 환경을 초기화

- C Shell (csh)

-C Shell 프로그램의 실행 명령어 : csh
-커맨드 구조에서 C 언어와 유사
-편리한 기능도 내장하고 있어서 많이 보급됨
-'.cshrc' 파일로 환경을 초기화

- Korn Shell (ksh)

-Bourne Shell과 완전히 호환
-'.kshrc' 파일로 환경을 초기화

- Bash Shell (bash)

-FSF(Free Software Foundation)에서 개발한 무료 공개용 쉘
-Korn Shell과 C Shell의 유용한 특징들을 통합
-대부분의 리눅스 기계에서의 표준적인 명령어 라인 인터페이스
-리눅스에서 처음 로그인을 하였을 때 기본으로 주어지는 쉘
-'.bashrc' 파일로 환경을 초기화

● 사용자 쉘 확인 및 변경
- /etc/passwd 파일 수정 예 (기본 쉘로 C Shell을 사용하도록 함)

nipark:x:502:502::/home/nipark:/bin/csh

◦ 해당 사용자의 로그인을 막으려고 할 때는 마지막 필드를 /bin/false로

- 사용하는 쉘을 확인하는 방법
  - echo $SHELL 명령을 수행
- 일시적으로 쉘을 변경하기
  - sh, csh, ksh, 또는 bash 명령을 수행
  - 원래의 쉘로 돌아오려면 exit 명령을 수행

- 쉘의 명령 완성 기능
- 명령어 일부만 입력하고 <Tab> 키를 누름
- 입력하지 않은 부분의 내용을 자동으로 생성해 줌
- 해당 명령어가 두 개 이상이면 명령어의 리스트를 모두 보여줌

- 쉘의 파일 이름 완성 기능
- 파일 이름의 일부만 입력하고 <Esc> 키를 누름
- 입력하지 않은 부분의 내용을 자동으로 완성해 줌
- 해당 파일이 두 개 이상이면 공통부분까지 만 완성

- 쉘의 History 기능
- 쉘 프롬프트에서 history 명령 또는 위, 아래 커서 키 사용
- ! history번호 <Enter> 또는 위/아래 화살표로 이동하여 <Enter>
- history 파일
  - ~/.bash_history (저장된 명령어 개수는 HISTSIZE 변수에 지정)
- 파일 이름에서 메타 문자의 지원
  - '?' : 임의의 한 문자, '*' : 문자 수와 상관없는 임의의 문자
  - 예: rm *.bak (파일들의 확장자가 'bak'으로 끝나는 파일 삭제

- 쉘에서 사용하는 특수 문자
- > : 표준 출력을 기록할 파일 지정
- ls > result
- >> : 표준 출력을 덧붙일 파일 지정
- < : 표준 입력을 읽을 파일 지정
- * : 0개 이상의 문자와 대응
- ? : 하나의 문자와 대응
- | : 어떤 프로세스의 출력을 다른 프로세스의 입력으로 보냄 (파이프)
  - 예: cat file1 file2 | sort | more
- ; : 명령 순서
- || : 이전 명령이 실패하면 실행하는 조건부 실행
- && : 이전 명령이 성공하면 실행하는 조건부 실행
- & : 명령을 백그라운드 프로세스로 실행
- # : 주석 처리
- $: 변수 접근 기호
  - 예: echo $PATH $LANG

● 쉘의 환경 설정

- 쉘 환경 설정 파일은 사용자의 홈 디렉터리에 존재
- 파일의 이름은 rc(Resource Configuration)
- 환경설정 파일의 종류
  - /etc/profile : 로그인 쉘로 로그인할 때 실행됨, 모든 사용자에게 영향
  - ~/.bash_profile : 로그인 쉘로 로그인할 때 실행됨. 개별 사용자가 설정
  - /etc/bashrc : 보통 ~/.bashrc에서 호출됨
  - ~/.bashrc : 새로운 쉘이 실행될 때 설정
  - ~/.bash_logout : 마지막 쉘이 종료되면서 실행

● 쉘의 환경 변수

- 주요 환경 변수

$HOME : 사용자 홈 디렉터리 $PATH : 명령어 프로그램 탐색 경로
$MAIL : 메일 보관함 경로 $MAILCHECK : 메일 확인 간격(초)
$SHELL : 현재 쉘의 경로$HOSTNAME : 컴퓨터 이름
$PWD : 현재 디렉터리 $PS1 : 프롬프트
$UID : 사용자 UID $USER : 사용자 이름
$HISTFILE : 히스토리 파일 $LANG : 사용 언어

- 환경 변수 값 확인
  - echo $환경변수
- 환경 변수 값을 모두 확인
  - printenv
- 환경 변수 값 설정
  - export [환경변수 [= 변수값]]
- 일반 변수 값 설정
  - 변수 = 변수값 (set 명령으로 값을 모두 확인)

● 다음 문제의 정답을 고르시오.

11. 다음 리눅스 명령어 중 bash 쉘의 내장 명령이 아닌 것은?
① cd ② alias
③ echo ④ rm

12. 다음 쉘의 기본 개념 설명 가운데 옳지 않은 것은?
① 쉘은 CPU를 직접 제어한다.
② 쉘은 커널과 통신한다.
③ 사용자가 입력한 내용은 디바이스 드라이버를 통과한다.
④ 쉘은 사용자의 지시를 수신한다.

14. BASH에서 다음 중 쉘에 의해 자동으로 설정되는 변수가 아닌 것은?
① PPID ② PWD

③ OSTYPE　　④ MYNAME

15. 다음 쉘 가운데 리눅스의 기본 쉘이며 가장 많이 사용되는 것은?
① csh　　② bash
③ kxh　　④ tcsh