8강. 프로세스 관리 명령(1)

● 파일 시스템 및 디스크 관리 명령
mount, umount, mkfs, fsck, fdisk, mkswap, du, df 등

● mount 명령의 기능
• 장치에 구성된 파일 시스템을 지정된 디렉터리(마운트 지점)에 붙여 파일 시스템을 사용할 수 있게 함
• 명령 형식
◦ mount -a [-fnrvw] [-t 파일시스템유형]
◦ mount [-fnrvw] [-o 옵션 [,...]] 장치 | 디렉터리
◦ mount [-fnrvw] [-t 파일시스템유형] [-o 옵션] 장치 디렉터리
• mount 관련 파일 : /etc/fstab
-부팅 시 마운트되는 파일 시스템 목록을 가짐
-예: 다음 줄을 /etc/fstab에 추가하고 /mnt/cdrom 디렉터리 생성함
◦ /dev/cdrom /mnt/cdrom iso9660 noauto,user,ro 0 0

● mount 명령의 사용법
• 옵션
◦ -f : 실제 마운트 하지 않고 마운트 가능한지 점검
◦ -r : 읽기 전용으로 마운트
◦ -w : 읽기, 쓰기 모드로 마운트
◦ -t type : 파일 시스템 유형을 type으로 지정
◦ -a : /etc/fstab에 나열된 모든 파일 시스템을 마운트
◦ -o options : 마운트 옵션을 지정, 옵션이 여러 개일 경우 ','로 구분

● mount 명령의 사용 예
◦ mount [-t ext3] (현재 마운트 된 [ext3 유형의] 모든 파일 시스템을 보여줌)
◦ mount -t iso9660 /dev/cdrom /mnt/cdrom (CD-ROM 드라이브를 /mnt/cdrom 디렉터리에 마운트)
◦ mount -a [-t iso9660] (/etc/fstab에 나열된 [iso9660 유형의] 모든 파일 시스템을 마운트)
◦ mount /dev/cdrom 또는 mount /mnt/cdrom (/etc/fstab에 기술된 파일 시스템을 마운트 할 경우 장치명 또는 마운트 포인트만 명시)
◦ mount /dev/sdb1 /media/usb (usb 장치를 /media/usb 디렉터리에 마운트)

● 운영체제 별 파일 시스템을 mount하는 예
• 해당 장치가 존재하는지, 파일 시스템 종류가 맞는지 확인
• mount 명령을 실행하기 전 마운트 지점 디렉터리를 생성

- 사용 예
  - mount -t msdos /dev/hda2 /mnt/msdos
  - mount -t vfat /dev/hda3 /mnt/win98
  - mount -t ntfs /dev/hda4 /mnt/win2000
  - mount -t ext2 /dev/hda5 /mnt/linux

- umount 명령
- 디렉터리에 마운트 되어 있는 저장 장치를 해당 디렉터리로부터 분리
- 명령 형식
  - umount -a [-nv] [-t 파일시스템유형]
  - umount [-nv] 장치 | 디렉터리
- 옵션
  - -a : /fstab에 나열된 모든 파일 시스템을 언마운트
  - -t type : 해당 파일 시스템 유형에만 적용
- umount 명령의 사용 예
  - umount /mnt/cdrom 또는 umount /dev/cdrom (현재 /mnt/cdrom 디렉터리에 마운트 되어 있는 CD-ROM 드라이브를 언마운트)
  - umount -a -t iso9660 (/etc/fstab에 나열되어 있는 파일 시스템 중 iso9660 유형을 모두 언마운트)
  - umount /media/usb (현재 /media/usb 디렉터리에 마운트 되어 있는 usb 장치를 언마운트)

- mkfs 명령
- 하드디스크 파티션에 리눅스 파일 시스템을 생성
- 파일 시스템의 생성 절차

-사용자는 root 권한을 가져야 함
-사용되지 않는 디스크나 파티션이 있어야 함
-VirtualBox에서 추가 디스크 생성 가능 (가상머신 전원 off 상태에서 [설정]-[저장소]-[컨트롤러:IDE]-마우스 오른쪽 버튼-[하드디스크 추가하기] 실행)
-실험용 USB를 준비하고 실습할 수 있음

- mkfs 명령의 사용법
- mkfs [-t 파일시스템유형] [fs-options] 장치이름 [블록]
- 옵션
  - -t : 만들어질 파일 시스템의 유형 지정
  - fs-options : 파일 시스템 옵션
  - 블록 : 파일 시스템을 위해 사용되는 블록의 개수
- mkfs 명령의 사용 예
  - mkfs -t ext3 /dev/hdb (하나의 파티션으로 이루어진 IDE primary slave 디스크에 ext3 파일 시스템을 생성)

- fsck 명령

• 기능 및 특징

-관리자 명령으로 마운트되어 있지 않은 파일 시스템을 검사
-파일 시스템의 일관성을 검사하고 대화식으로 파일 시스템을 복원
-명령 수행 시 유효한 데이터가 유실될 가능성이 있는 조작은 피함

• 명령 형식

◦ fsck [-AVRTNP] [-s] [-t 파일시스템유형] [옵션] [파일시스템]

• 관련 파일

/etc/fstab, /etc/filesystems

- fdisk 명령

• 하드 디스크 파티션을 관리하는 대화식 유틸리티
• 명령 형식

◦ fdisk [옵션] [하드디스크장치명]

• fdisk 대화식 명령(메뉴 방식)의 종류

◦ p : 파티션 테이블 출력
◦ n : 새로운 파티션 추가
◦ d : 파티션 삭제
◦ w : 파티션 테이블 기록하고 끝냄

• 사용 예

◦ tail -f /var/log/messages로 확인하고 <Ctrl>+<C> (usb 장치 이름을 확인)
◦ umount /dev/sdb1
◦ mkdir /media/usb1; mkdir /media/usb2
◦ fdisk -cu /dev/sdb

p로 확인하고 d로 지움

n p 1하고 p로 확인, n p 2하고 p로 확인 후, w

◦ mkfs -t vfat /dev/sdb1; mkfs -t vfat /dev/sdb2
◦ mount /dev/sdb1 /media/usb1; mount /dev/sdb2 /media/usb2

- mkswap 명령

• 지정한 특정 장치나 파일을 리눅스용 스왑 영역으로 지정
• mkswap 명령 형식

◦ mkswap [옵션] 장치또는파일명 [블록크기]
◦ 장치이름 인자로 사용될 수 있는 장치이름 : /dev/hda[1-8], /dev/hdb[1-8], /dev/sda[1-8], /dev/sdb[1-8]
◦ 블록크기는 원하는 블록 단위 크기

• swapon / swapoff 명령

◦ 스왑 영역 사용 설정 / 해제
◦ free -m으로 스웝 영역을 확인함

- mkswap 명령 사용 예

- 스왑 파일 사용의 일반적 처리 과정
  - dd if=/dev/zero of=/swapfile bs=1024 count=8192
  - mkswap /swapfile 8192
  - swapon /swapfile (swap 영역으로 사용됨)
  - swapoff /swapfile (swap 영역 사용이 해제됨)

- du(disk usage) 명령
- 파일에 사용되는 블록 수를 표시
- 명령 형식

du [옵션] [디렉터리...]

- 사용 예
  - du
  - du -s (하위 디렉터리 내역을 생략하고 총 블록 수만 표시)
  - du --max-depth=1 (바로 아래 디렉터리까지만 블록 수 표시)
  - du /home (/home 디렉터리와 하위 디렉터리의 블록 수 표시)

- df(disk free) 명령
- 지정한 파일이 있는 파일 시스템의 디스크 공간 정보를 보여줌
- 명령 형식

df [옵션] [파일명...]

  - 사용 예
  - df (모든 파일 시스템의 공간 정보)
  - df -i (모든 파일 시스템의 i-node 사용 정보)
  - df /home (/home 디렉터리가 있는 파일 시스템의 공간 정보)

- quota 명령
- 사용자의 디스크 사용량 한도를 설정
- 관련된 명령들
  - quota [옵션] user | group : 사용량 및 한도 표시
  - quotacheck : quota 검사 및 정보 생성
  - edquota : 사용자, 그룹별 사용량 한도 설정
  - quotaon : 사용량 한도 적용
  - quotaoff : 사용량 한도 해제
- quota 명령 사용 예
  - VirtualBox에서 IDE primary slave 디스크 추가
  - fdisk 명령으로 파티션을 생성하지 않으면 하나의 파티션이 존재
  - mkfs -t ext3 /dev/hdb (/dev/hdb 디스크의 파티션에 ext3 파일 시스템 생성)
  - mkdir /mnt/linux (마운트 지점 /mnt/linux 디렉터리 생성)
  - /etc/fstab 파일에 다음 줄 추가

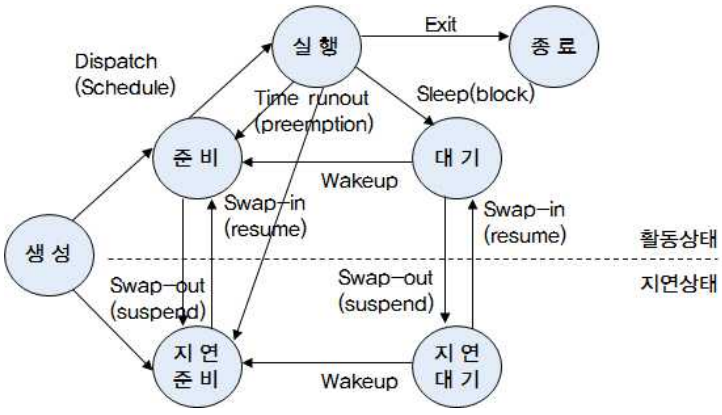
```
/dev/hdb    /mnt/linux    ext3    defaults,usrquota    1    2
```

◦ mount /dev/hdb (/etc/fstab의 내용대로 파티션을 /mnt/linux 디렉터리에 마운트)
◦ quotacheck -cug /mnt/linux (quota 관리 파일이 /mnt/linux에 생성됨)
◦ edquota -f /mnt/linux nipark (nipark 사용자의 quota를 편집하여 지정)
◦ quotaon /mnt/linux (quota 기능을 시작)
◦ quota nipark (nipark 사용자의 사용량을 보여줌)

● 프로세스(process) 의 정의
• 실행되고 있는 프로그램
• 커널에 등록되어 관리를 받는 작업
• 커널은 프로세스 관리 블록(PCB, Process Control Block)에 정보를 저장

● 프로세스 관리 블록 개요
커널에 등록된 각 프로세스에 대한 정보를 저장하는 영역
• 프로세스들은 모두 커널 공간에 자신의 PCB를 가지며 커널이 PCB를 관리함
• 프로세스의 개념과 작동 레벨
• 프로세스 고유 번호(PID)
-커널이 시스템 내의 프로세스마다 고유하게 부여하는 번호
-프로세스가 생성 시 부여
• 프로세스의 우선순위(priority)
-프로세스 스케줄링을 위한 정보
-프로세서를 할당할 프로세스를 선정하는데 참조
• 현재 상태(current state)
프로세스가 할당 받은 자원과 현재 상태
• 프로세스가 할당 받은 자원에 대한 정보
-어느 자원이 어느 프로세스에 할당되는지 알 수 있음
-자원 관리에 참고
• 문맥 저장 영역(context save area)
-프로세스가 실행을 중지해야 할 때 각종 레지스터(register)들의 값을 보관하기 위하여 사용
-해당 프로세스가 다시 실행될 때 레지스터의 값을 복원

● 프로세스 상태 전이도



● 프로세스의 상태 개요

• 활동(active) / 지연(suspended) 상태

-활동 상태 : 프로세스가 기억 장치를 할당 받은 상태

-지연 상태에서 resume 또는 swap-in 으로 활동 상태가 됨

-지연 상태 : 프로세스가 기억 장치를 할당 받지 못한 상태

-활동 상태에서 suspend 또는 swap-out 으로 지연 상태가 됨

• 준비 (ready) / 대기 (blocked) 상태

-준비 상태 : 실행에 필요한 모든 자원을 할당 받은 상태

-대기 상태 : 실행에 필요한 자원을 요청하고 기다리는 상태

• 생성(created) 상태

커널 공간에 PCB 등이 만들어지고 프로세스가 처음 생성되는 상태

• 준비(ready) 상태

-기억 장치 등 필요한 모든 자원을 할당 받은 상태에서 프로세서를 할당 받으려고 대기하는 상태

-프로세서를 할당 받게 되면 즉시 실행이 가능

-디스패치(dispatch) 또는 스케줄(schedule) : 준비 상태에서 실행 상태로 전이되는 것

• 실행(running) 상태

-프로그램 코드가 프로세서에 의해 실행되고 있는 상태

-프로세스가 필요한 모든 자원을 할당 받은 상태

-선점(preemption) : 실행 상태의 프로세스가 프로세서 시간 할당량이 끝나거나 우선순위가 높은 프로세스가 들어왔을 때 프로세서를 반납하고 준비 상태로 전이되는 것

-시간 종료(time runout) : 시간 할당량의 종료로 선점되는 경우

• 대기(block) 상태

-실행 상태의 프로세스가 자원을 요청하여 대기 상태로 전이되는 것

-프로세스가 필요한 자원을 요청하고 이를 할당 받을 때까지 기다리는 상태

-웨이크업(wakeup) : 프로세스에 요청한 자원이 할당되어 준비 상태로 전이되는 것

• 지연 대기 상태

프로세스가 대기 상태에서 기억 장치를 잃은 경우 지연 대기 상태로 전이됨

• 지연 준비 상태

-프로세스가 기억장치를 제외한 다른 모든 필요한 자원들을 보유한 상태

-지연 준비 상태로 전이되는 경우

◦ 생성 상태의 프로세스가 기억 장치 공간이 부족하여 전이

◦ 준비 상태의 프로세스가 기억 장치를 반납하고 전이

◦ 실행 상태의 프로세스가 선점 당할 때 기억 장치까지 반납하고 전이

● 리눅스 프로세스의 작동 레벨

리눅스 시스템의 실행 레벨(runlevel) : 0~6

● 다음 문제의 정답을 고르시오.

11(교재 6장). 파일 시스템 관리를 위한 명령어가 아닌 것은?

① fsck ② fdisk

③ mkfs ④ reiserfs

12(교재 6장). 파일 시스템의 이상 유무를 체크하기 위하여 fsck 명령을 수행하였다. 검사의 출력 결과 값이 4가 나왔을 때 파일 시스템은 어떠한 상태인가?

① 파일 시스템이 고쳐지지 않은 에러가 남아 있음

② 에러 없음

③ 파일 시스템이 재부팅 필요

④ 파일 시스템 에러가 고쳐짐

13(교재 7장). 프로세스의 작동 레벨 중 재실행 모드로 거의 실행 레벨 0과 같지만, init.d가 시스템이 재실행될 것인지, 종료될 것인지를 결정하는 레벨은?

① Runlevel 1 ② Runlevel 3

③ Runlevel 4 ④ Runlevel 6

● 다음 문제에 대한 정답을 서술하시오.

16(교재 7장). PCB(Process Control Block)에 대하여 설명하고, PCB에 담긴 프로세스의 정보를 조사하시오.

(답) 프로세스 고유 번호(PID)는 커널이 시스템 내의 프로세스들을 관리하는 데 편리성을 가지려고 프로세스마다 고유하게 부여되는 번호이며, 이는 프로세스가 생성될 때에 부여된다. 프로세스의 우선순위(priority)는 운영체제가 프로세스 스케줄링을 하기 위한 정보이며, 여러 가지 프로세스들 가운데 프로세서를 할당할 프로세스를 선정하기 위한 우선순위를 참조하는 데 사용된다. 커널에 등록된 프로세스들이 각각 어떠한 자원들을 할당받고 있고 어떠한 상태에 있는가에 따라 현재 상태(current state)를 구분할 수 있으며, 이에 대한 정보가 프로세스 현재 상태 필드에 저장되고 프로세스 상태가 변화될 때마다 정보를 갱신하게 된다. 프로세스가 할당받은 자원들에 대한 정보를 저장해 놓는 영역 또는 포인터를 통하여 시스템 내의 어느 자원이 어느 프로세스에 할당되었는지 알 수 있으며, 이를 자원 관리에 참고하여 사용할 수 있다.

마지막으로 문맥 저장 영역(context save area)은 프로세스가 실행 도중 어떤 이유 때문에 실행을 중지해야 할 때 프로세서의 각종 레지스터(registers)에 저장하여 갖고 있던 값을 보존하기 위하여 사용된다. 즉, 실행중 인 프로세스가 중지되게 되면 레지스터 문맥을 PCB 영역 내에 저장하게 되고, 이 프로세스가 다시 실행될 때에는 PCB 내에 저장되었던 내용을 다시 프로세서내의 레지스터에 되돌려 놓는 일을 한다.

17(교재 7장). 프로세스 상태 전이도를 그려보고 각각의 상태에 대하여 설명하시오.

① 생성 상태: 사용자가 요청한 작업이 커널에 등록되어 커널 공간에 PCB 등이 만들어지고 프로세스가 처음 생성되는 상태

② 준비 상태: 프로세스가 기억 장치를 비롯한 필요한 모든 자원을 할당받은 상태에서 프로세서를 할당받으려고 대기하는 상태

③ 실행 상태: 프로세스의 프로그램 코드가 기억 장치로부터 읽히면서 프로세서에 의해 실행되고 있는 상태
④ 대기 상태: 프로세스가 임의의 자원을 요청하고 이를 할당받을 때까지 기다리는 상태
⑤ 지연 상태: 프로세스가 기억 장치를 할당받지 못한 상태
⑥ 지연 준비 상태: 프로세스는 다른 모든 필요한 자원들을 보유하고는 있지만, 기억 장치와 프로세서를 잃은 상태
⑦ 지연 대기 상태: 대기 상태에서 기억 장치를 잃은 프로세스의 상태