

# 제9장 유틸리티

숙명여대 창병모

## 9.1 명령어 스케줄링

# 주기적 실행 cron

---

- cron 시스템
  - 유닉스의 명령어 스케줄링 시스템으로
  - crontab 파일에 명시된 대로 주기적으로 명령을 수행한다.
- crontab 파일 등록법

\$ crontab 파일

crontab 파일을 cron 시스템에 등록한다.

- crontab 파일
  - 7개의 필드로 구성
  - 분 시 일 월 요일 [사용자] 명령

# 주기적 실행 cron

---

- crontab 명령어

```
$ crontab -l [사용자]
```

사용자의 등록된 crontab 파일 리스트를 보여준다.

```
$ crontab -e [사용자]
```

사용자의 등록된 crontab 파일을 수정 혹은 생성한다.

```
$ crontab -r [사용자]
```

사용자의 등록된 crontab 파일을 삭제한다.

# crontab 파일 예

---

- chang.cron

---

```
30 18 * * * rm /home/chang/tmp/*
```

---

- 사용예

```
$ crontab chang.cron
```

```
$ crontab -l
```

```
30 18 * * * rm /home/chang/tmp/*
```

```
$ crontab -r
```

```
$ crontab -l
```

```
no crontab for chang
```

# crontab 파일 예

---

- crontab 파일 예1

```
0 * * * * echo "빠꼭" >> /tmp/x
```

매 시간 정각에 "빠꼭" 메시지를 /tmp/x 파일에 덧붙인다.

- crontab 파일 예2

```
20 1 * * * root find /tmp -atime +3 -exec rm -f {} \;
```

매일 새벽 1시 20분에 3일간 접근하지 않은 /tmp 내의 파일을 삭제

- crontab 파일 예3

```
30 1 * 2,4,6,8,10,12 3-5 /usr/bin/wall /var/tmp/message
```

2개월마다 수요일부터 금요일까지 1시 30분에 wall 명령을 사용해서  
시스템의 모든 사용자에게 메시지를 전송

# 한번 실행: at

---

- at 명령어
  - 미래의 특정 시간에 지정한 명령어가 한 번 실행되도록 한다.
  - 실행할 명령은 표준입력을 통해서 받는다.

- 사용법

`$ at [-f 파일] 시간`

지정된 시간에 명령이 실행되도록 등록한다. 실행할 명령은 표준입력으로 받는다.

-f: 실행할 명령들을 파일로 작성해서 등록할 수도 있다.

- 예

```
$ at 1145 jan 31
```

```
at> sort infile > outfile
```

```
at> <EOT>
```

# 한번 실행: at

---

- atq 명령어
  - at 시스템의 큐에 등록되어 있는 at 작업을 볼 수 있다

- 사용예

\$ atq

```
Rank Execution Date Owner Job Queue Job Name
1st Jan 31, 2012 11:45 chang 1327977900.a a stdin
```

- at -r 옵션

\$ at -r 작업번호

지정된 작업번호에 해당하는 작업을 제거한다.

- 사용 예

\$ at -r 1327977900.a



## 9.2 디스크 및 아카이브

# 디스크 사용: df

---

- df 명령어
  - 파일시스템에 대한 정보를 보여준다.

\$ df 파일시스템\*

파일 시스템의 사용중이거나 사용 가능한 디스크 공간에 대한 정보를 보여준다.

- 사용 예

\$ df

Filesystem	1K-blocks	Used	Available	Use%	Mounted on
/dev/mapper/root	51606140	7570736	41413964	16%	/
tmpfs	1030972	676	1030296	1%	/dev/shm
/dev/sda1	495844	29048	441196	7%	/boot
/dev/mapper/home	424544656	3577668	399401344	1%	/home

# 디스크 사용: du

---

- du 명령어

```
$ du [-s] 파일*
```

파일이나 디렉토리가 사용하는 디스크 사용량(블록 수)을 알려준다.

- 파일을 명시하지 않으면 현재 디렉터리의 사용 공간을 보여준다.

- 사용 예

```
$ du
```

```
258 ./htdocs/images
```

```
42 ./htdocs/lecture/math
```

```
2582 ./htdocs/lecture/sp/lab
```

```
...
```

- 사용 예

```
$ du -s
```

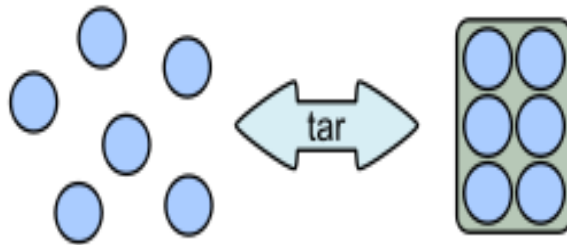
```
-s(sum)
```

```
22164 .
```

# tar 아카이브

---

- 아카이브
  - 백업 또는 다른 장소로의 이동을 위해 여러 파일들을 하나로 묶어놓은 묶음
  - 아카이브를 만들거나 푸는데 tar(tape archive) 명령어 사용
- tar의 역학



# tar 아카이브

---

- tar 명령어

- 옵션: c(create), v(verbose), x(extract), t(table of contents), f(file)
- `$ tar -cvf 타르파일 파일+`  
여러 파일들을 하나의 타르파일로 묶는다. 보통 확장자로 .tar 사용
- `$ tar -xvf 타르파일`  
하나의 타르파일을 풀어서 원래 파일들을 복원한다.
- `$ tar -tvf 타르파일`  
타르파일의 내용을 확인한다.

# tar 아카이브: 사용 예

---

- 현재 디렉터리에 있는 모든 파일을 다른 곳으로 옮기기

```
$ tar -cvf src.tar *
```

... src.tar를 다른 곳으로 이동

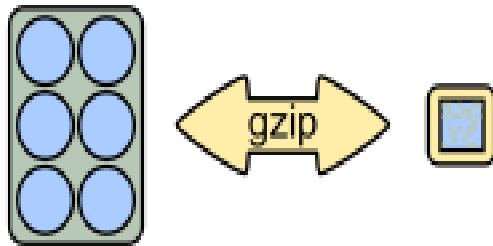
```
$ tar -tvf src.tar
```

```
$ tar -xvf src.tar
```

## 9.3 파일 압축

# 파일 압축: gzip

- gzip 명령어



\$ gzip [옵션] 파일\*

파일(들)을 압축하여 .gz 파일을 만든다.

-d : 압축을 해제한다.

-l : 압축파일 안에 있는 파일 정보(압축된 크기, 압축률) 출력한다.

-r : 하위 디렉터리까지 모두 압축한다.

-v : 압축하거나 풀 때 압축률, 파일명을 출력한다.



# 압축 풀기

---

- 사용법

```
$ gzip -d 파일.gz*
```

gzip으로 압축된 파일들을 복원한다.

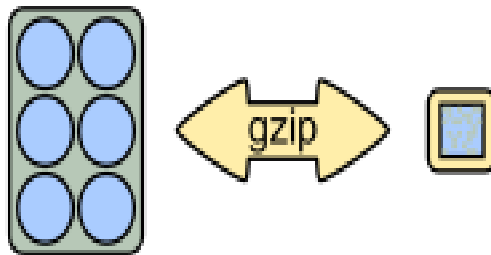
```
$ gunzip 파일.gz*
```

gzip으로 압축된 파일들을 복원한다.

# 파일 압축: gzip

---

- gzip 명령어



- 사용법

\$ gzip 파일\*

\$ gzip -d 파일.gz\*

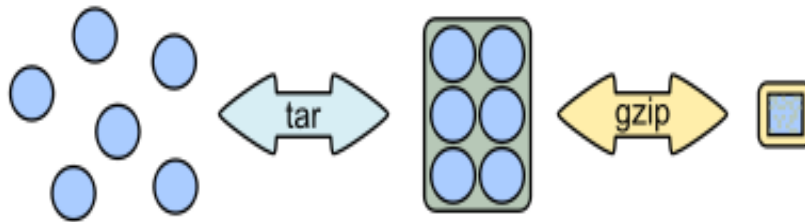
- 사용 방법

- 파일들을 하나의 타르파일로 묶은 후 compress/gzip을 사용해 압축
- 파일 복원: 압축을 해제한 후, 타르파일을 풀어서 원래 파일들을 복원

# 사용 예

---

- 사용 예
  - 파일들을 하나의 타르파일로 묶은 후 gzip을 사용해 압축
  - 파일 복원: 압축을 해제한 후, 타르파일을 풀어서 원래 파일들을 복원



```
$ tar -cvf src.tar *
```

```
$ gzip src.tar
```

... 이 파일을 원하는 곳으로 이동

```
$ gzip -d src.tar.gz
```

```
$ tar -xvf src.tar
```

# 파일 압축: compress

---

- 명령어 compress/ uncompress 명령어

\$ compress 파일\*

파일(들)을 압축하여 .Z 파일을 만든다.

\$ uncompress 파일.Z\*

압축된 파일(들)을 복원한다.

- 사용 예

```
$ ls
```

```
304 -rw-r--r-- 1 chang faculty 143360 Oct 8 2012 src.tar
```

```
$ compress src.tar
```

```
$ ls
```

```
54 -rw-r--r-- 1 chang faculty 27422 Oct 8 2012 src.tar.Z
```

```
$ uncompress src.tar.Z
```

```
$ ls
```

```
304 -rw-r--r-- 1 chang faculty 143360 Oct 8 2012 src.tar
```

# RPM(Red Hat Package Manager)

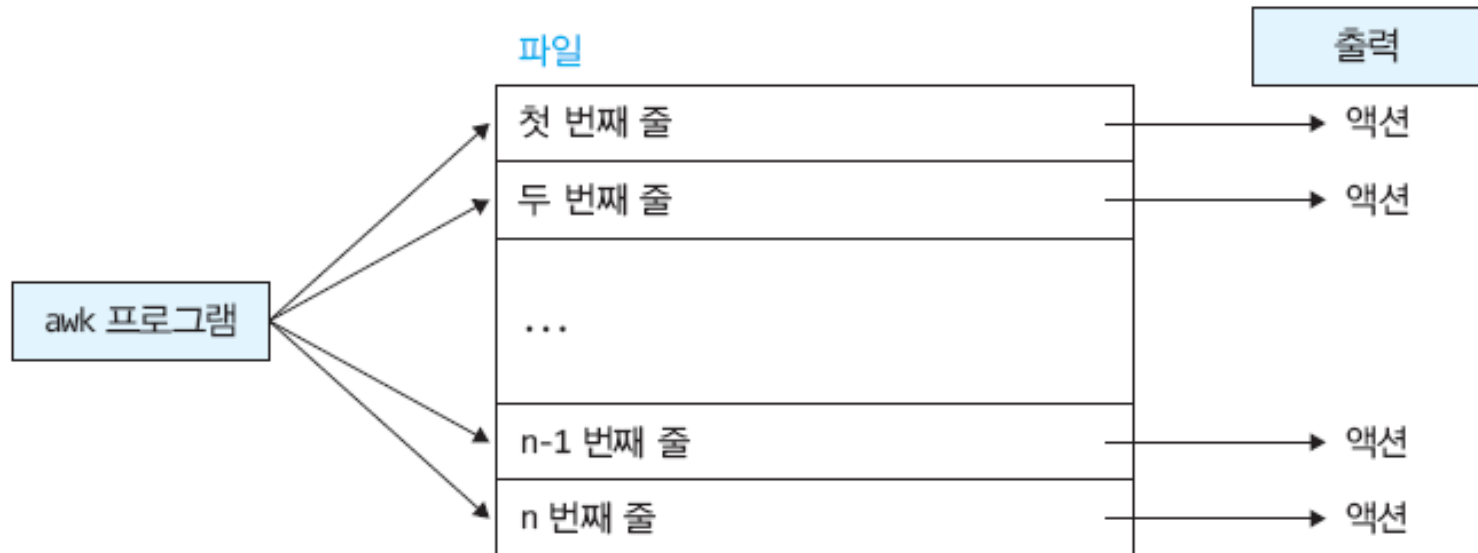
---

- RPM 패키지 매니저
  - 원래 레드햇에서 사용되었던 패키지 파일 및 관리 소프트웨어
  - 현재는 Linux Standard Base의 표준 패키지 포맷 중 하나이다.
  - 각종 소프트웨어의 설치 및 업데이트를 편리하게 할 수 있다.
- 
- 소프트웨어 설치 및 업데이트
  - `rpm -Uvh foo-1.0-1.i386.rpm`
- 이미 설치된 패키지 대치
  - `rpm -ivh --replacefiles foo-1.0-1.i386.rpm`
- 패키지 제거하기
  - `rpm -e foo-1.0-1.i386`

## 9.4 AWK

# AWK

- AWK
  - 일반 스크립트 언어
  - AWK(Aho, Weinberger, Kernighan)
  - 텍스트 형태로 되어있는 각 줄을 필드로 구분하여 처리한다.
  - 필드: 줄을 구성하는 단어



# AWK

---

- awk 프로그램

- 간단한 프로그램은 명령줄에 직접 작성하여 수행
- awk 프로그램을 파일로 작성하여 -f 옵션을 이용하여 수행

```
$ awk 프로그램 파일*
```

```
$ awk [-f 프로그램파일] 파일*
```

텍스트 파일을 대상으로 하여 각 줄을 필드들로 구분하고 이들을 awk 프로그램이 지시하는 대로 처리한다.



# awk 프로그램

---

- awk 프로그램
  - 조건과 액션을 기술하는 명령어들로 구성됨
  - [ 조건 ] [ { 액션 } ]
  - 대상 파일의 각 줄을 스캔하여 조건을 만족하는 줄에 액션 수행
- 간단한 awk 프로그램 예
  - \$ awk '{ print NF, \$0 }' you.txt
  - \$ awk '{ print \$1, \$3, \$NF }' you.txt
  - \$ awk 'NR > 1 && NR < 4 { print NR, \$1, \$3, \$NF }' you.txt

# 조건(condition)

---

- 조건에서 사용 가능한 연산자 및 패턴
  - BEGIN  
파일 시작
  - END  
파일 끝
  - 관계 연산자 혹은 논리 연산자를 포함한 조건식
  - /패턴/  
패턴에 해당하는 줄
  - 패턴1, 패턴2  
패턴1을 포함한 줄부터 패턴2를 포함한 줄까지

# 액션(action)

---

- 액션에서 사용 가능한 문장
  - if (조건) 실행문 [else 실행문]
  - while (조건) 실행문
  - for (식; 조건; 식) 실행문
  - break
  - continue
  - 변수 = 식
  - print [식들의 리스트]
  - printf 포맷 [, 식들의 리스트]
  - next  
현재 줄에 대한 나머지 패턴 건너뛰기
  - exit  
현재 줄의 나머지 부분 건너뛰기
  - { 실행문 리스트 }

# 연산자

---

- 액션에서 사용 가능한 연산자(C 언어 연산자)
  - 산술 연산자: +, -, \*, /, %, ++, --
  - 대입 연산자: =, +=, -=, \*=, /=, %=
  - 조건 연산자: ? :
  - 논리 연산자: ||, &&, !
  - 패턴 비교 연산자: ~, !~
  - 비교 연산자: <, <=, >, >=, !=, ==
  - 필드참조 연산자: \$

# 간단한 AWK 프로그램 예

---

- `$ awk 'END { print NR }' 파일이름`
- `$ awk 'NR % 2 == 0 { print $0 }' 파일이름`
- `$ awk 'NF > 5 { print $0 }' 파일이름`
- `$ awk '/raise/ { print $0 }' 파일이름`
- `$ awk '/the?/ { print $0 }' 파일이름`
- `$ awk '/a..e/ { print $0 }' 파일이름`
- `$ awk '/a.*e/ { print $0 }' 파일이름`
- `$ ls -l | awk '{x += $5}; END {print x}'`

## 9.5 AWK 프로그램 작성

# AWK 프로그램 예

---

- [예제 1]

```
BEGIN { print "파일 시작:", FILENAME }  
{ print $1, $NF }  
END { print "파일 끝" }
```

- [예제 2]

```
BEGIN { print "파일 시작" }  
{  
  printf "line %d: %d \Wn", NR, NF;  
  line++;  
  word += NF  
}  
END { printf "줄 수 = %d, 단어 수 = %d\Wn", line, word }
```

# AWK 프로그램 예

---

- [예제 3]

```
{  
    for (I = 1; I <= NF; I += 2)  
        printf "%s ", $I  
        printf " \n"  
}
```
- [예제 4]

```
/st.*e/ {print $0 }
```
- [예제 5]

```
/strong/, /heart/ { print $0 }
```



# AWK 프로그램 예

---

- [예제 6]

```
/raise/ { ++line }
```

```
END { print line }
```

- [예제 7]

```
BEGIN {
```

```
    FS="[ ^a-zA-Z]+"
```

```
}
```

```
{
```

```
    for (i=1; i<=NF; i++)
```

```
        words[tolower($i)]++
```

```
}
```

```
END {
```

```
    for (i in words)
```

```
        print i, words[i]
```

```
}
```

# 핵심 개념

---

- cron은 유닉스의 명령어 스케줄링 시스템으로 crontab 파일에 명시된 대로 주기적으로 명령을 수행한다.
- 유닉스에서는 tar 명령어를 사용하여 여러 파일을 하나로 묶은 후에 compress 혹은 gzip 명령어를 이용하여 압축한다.
- awk 프로그램은 조건과 액션을 기술하는 명령어들로 구성되며 텍스트 파일의 줄들을 스캔하여 조건을 만족하는 각 줄에 대해 액션을 수행한다.