7. 기타 시스템 관리

Diff

목차

- Cron
  - `crond`, `crontab`
- System Logs
  - `syslogd`, `logrotate`, `logcheck`, `fail2ban`

만약..

- 어떤 일을 일정 시각마다 반복하게 하려면?
- 2012-06-14 04:00 디스크에 있는 파일을 백업한다.
- 2012-06-15 04:00 디스크에 있는 파일을 백업한다.
- 2012-06-16 04:00 디스크에 있는 파일을 백업한다.
- 2012-06-17 04:00 디스크에 있는 파일을 백업한다.
- 2012-06-18 04:00 디스크에 있는 파일을 백업한다.
- 2012-06-19 04:00 디스크에 있는 파일을 백업한다.
- 2012-06-20 04:00 디스크에 있는 파일을 백업한다...
- 이걸 사람이 꼬박 꼬박 해줄 수는 없다

  → 이런 걸 대신 해 주는게 cron

`crond`

- Cron은 명령어나 쉘 스크립트 따위의 일을 스케줄 할 수 있게 해준다.
- 지정된 시각과 날짜에 주어진 일을 하게 할 수 있다.
- `crond`(cron daemon)이라는 놈이 이런 일들을 처리해준다.

  - Daemon(데몬) : 대부분의 시간에 잠들어 있고 필요할 때마다 일하는 프로그램.
- 잘 일하고 있나 확인하기 : `ps aux | grep cron`
- cron 데몬이 일을 할 때에는 우선 해야 할 일들의 목록을 얻어와야 한다.

  ... 어디에서?

`crontab`

- *cron tab*le의 약자.
- 언제 무슨 일을 할 것인지 `crond`에게 알려주는 파일.
- `cd /etc; ls | grep crond`?
- `/etc/crontab` : 시스템 cron의 설정파일
- `/etc/cron.hourly`, `daily`, … : 매시, 매일, …
- 하지만 저 파일은 오직 root만이 직접 수정할 수 있다.
- 다행히 cron은 여러 사용자들이 동시에 쓸 수 있도록 만들어졌다.
- `/var/spool/cron/crontabs/사용자이름`에 자신의 설정파일을 만들면 된다.
- 하지만 일반 사용자는 crontabs 디렉토리의 열람 권한조차 없다! 어떻게 해야 될까?

`crontab` 명령

- 자기 자신의 crontab 파일은 직접 편집하는게 아니라, `crontab` 명령을 이용해야 한다.
- `crontab -e` : 자신의 crontab 편집하기.
- `crontab <파일이름>` : 해당하는 파일로 crontab 교체하기.
- `crontab -l` : 자신의 crontab 목록 보기.
- `crontab -r` : 자신의 crontab 지우기.
- 적절한 권한이 있다면 다른 사용자의 crontab도 건드릴 수 있다.
- `crontab -u <다른 사용자 아이디> -l` : 다른 사용자의 crontab 목록 보기.
- 그런데, crontab 파일은 어떻게 생겼지?

vi /etc/crontab

differ@sparcs: ~

```
# /etc/crontab: system-wide crontab
# Unlike any other crontab you don't have to run the `crontab'
# command to install the new version when you edit this file
# and files in /etc/cron.d. These files also have username fields,
# that none of the other crontabs do.

SHELL=/bin/sh
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin

# m h dom mon dow user  command
17 *    * * *   root    cd / && run-parts --report /etc/cron.hourly
25 6    * * *   root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.daily )
47 6    * * 7   root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.weekly )
52 6    1 * *   root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.monthly )
#
~
"/etc/crontab" [읽기 전용] 15L, 722C                                    1,1          모두
```

- 주석 : 맨 앞 글자가 #인 줄은 무시된다.
  그럼 가운데 #자를 넣으면? 그건 주석이 아니다.
- 비어있는 행과 행 앞의 공백 문자는 무시.
- 환경변수를 설정해주고...
- cron에 지시할 것들을 적어주면 끝.

환경변수

- `이름 = 값`으로 설정할 수 있다.
- 등호 양 옆을 제외하고 `값`에 들어있는 공백은 `값`에 포함된다.
- `값` 앞에 공백을 넣으려면 ""나 ''로 감싼다.
- `값`은 환경변수로 치환되지 않는다. `PATH = $HOME/bin:$PATH` 같은 건 불가능.
- `SHELL` : cron이 돌아가는 shell. 기본값 `/bin/sh`
- `EDITOR` : 편집기. 기본값 `vim`
- `PATH` : cron에 쓰일 프로그램을 탐색할 경로.
- `HOME` : cron에 쓰일 홈 디렉토리.
- `LONGNAME` : 해당 crontab의 소유주
- `MAILTO` : cron 출력을 받아볼 이메일. 기본적으로 소유주의 이메일로 세팅되어 있다.

cron 명령어

- 분 시 날짜 월 요일 년도 유저 명령 와 같이 한 줄에 하나씩 지정해준다.
- 년도와 유저는 필수가 아니다. 하지만 SPARCS 서버의 crontab에서는 유저는 지정해주게 되어있다.
- 모든 필드의 조건이 만족되었을 때에만 명령이 실행된다.
- 분은 0~59, 시는 0~23까지의 값을 가진다.
- 날짜는 1~31, 월은 1~12 또는 JAN~DEC까지의 값을 가진다.
- 요일은 0~6 또는 SUN~SAT, 년도는 1970~2099까지의 값을 가진다.

cron 명령어 - 특수문자

- `*` : 모든 값을 의미한다. 예를 들어 `월`에 `*`를 적었다면 "매달"을 의미한다.
- `-` : 범위를 나타낼 때 쓴다. 예를 들어 `요일`에 `MON-FRI`를 적었다면 "월요일부터 금요일까지"를 의미한다.
- `/` : 띄엄 띄엄 떨어진 값을 나타낼 때 쓴다. 예를 들어 `분`에 `4-44/4`를 적었다면 "4분부터 44분까지 4분 간격"을 의미한다.
- `W`, `#` : 각각 '제일 가까운 주중', 'n번째 x요일'을 의미할 때 쓰이나 많이 쓸 것 같지는 않다.

cron 명령어 - 예제

differ@sparcs: ~

```
# /etc/crontab: system-wide crontab
# Unlike any other crontab you don't have to run the `crontab'
# command to install the new version when you edit this file
# and files in /etc/cron.d. These files also have username fields,
# that none of the other crontabs do.

SHELL=/bin/sh
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin

# m h dom mon dow user  command
17 *    * * *   root    cd / && run-parts --report /etc/cron.hourly
25 6    * * *   root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.daily )
47 6    * * 7   root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.weekly )
52 6    1 * *   root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.monthly )
#
~
"/etc/crontab" [읽기 전용] 15L, 722C                                    1,1          모두
```

- `* * * * * sl` : 매분 `sl`을 실행한다.
- `05 * * * * sl` `05 20 * * * sl` : 매시 5분 `sl`을, 매일 오후 8시 5분 `sl`을 실행한다.
- `*/5 * * * * sl` `03-59/5 * 10 * * sl` : 매 5분 `sl`을, 3분부터 매 5분 `sl`을 실행한다.
- `* 4 * * Sun sl` : 매주 일요일 오전 4시마다 `sl`을 실행한다.
- `00 4 * * Sun sl` `00 4 * * 0 sl` : 둘 다 같은 뜻이다.

미리 정의된 키워드들

- `@yearly`, `@annually` : 매년 1월 1일 자정에 실행한다.
- `@monthly` : 매달 1일 자정에 실행한다.
- `@weekly` : 매주 일요일 자정에 실행한다.
- `@daily` : 매일 자정에 실행한다.
- `@hourly` : 매시 정각에 실행한다.
- 위의 키워드 대신 쓸 수 있는 cron 명령어는?
- `@reboot` : 부팅되었을 때 실행한다.

`crontab` 퍼미션

- `/etc/cron.allow`와 `/etc/cron.deny`로 접근 권한을 설정한다.
- 한 줄에 사용자 ID 하나씩을 적는다.
- `cron.allow`가 존재한다면 (root 포함) 그 목록에 존재하는 사용자만 `crontab` 사용 가능.
- `cron.allow`가 존재하지 않고, `cron.deny`만 존재한다면, 그 목록에 존재하는 사용자를 제외하고 `crontab` 사용 가능.
- 둘 다 존재하지 않는다면 오직 root만이 `crontab` 사용 가능.
- 초기에는 둘 다 존재하지 않으므로, 필요할 때 생성한다.
- 문제 : 만약 `cron.allow`와 `cron.deny` 모두에 `differ`라는 사용자가 적혀있다면?

`anacron`

- 서버가 아닌 일반 컴퓨터의 경우 하루 24시간 내내 켜져 있지 않을 수 있다!
- 예를 들어 `0 0 1 * * backup` 같은 일이 있을 때, 1일 자정에 컴퓨터를 꺼 놨다면?
- `anacron` : 24시간 켜져있지 않은 개인용 컴퓨터에서, 일 이상의 간격으로 정해진 작업을 수행할 때 사용한다.

  예: 개인용 컴퓨터에서 사용되지 않는 파일의 삭제 등 집안일을 처리할 때..
- 여하튼 스팍스 서버에서는 딱히 쓰일 일이 없다.

cron 연습

- 홀수 분마다 현재 시각을 `date` 명령을 이용하여 출력시켜라.
- 결과는 홈 디렉토리의 `date.log` 파일에 저장시킨다.

로그?

영어 **한국어** 일본어

로그

New! 대체 번역을 보거나 수정하려면 위에서 단어를 클릭하세요. 무시

명사

로그	log, logarithm
통나무	log
공정 일지	log
항해 일지	logbook, log
측정기	measuring instrument, log
항공 일지	logbook, log
엔진 따위의 공정 일지	log

동사

기록하다	record, log, enroll, write down, score, memorize
통나무로 자르다	log
공정 따위를 기록하다	log
...의 항정을 기입하다	log

형용사

통나무로 만든	log

시스템 로그

- 시스템이 켜질 때부터 꺼질 때까지의 모든 상황을 프로그램이 기록하여 파일로 저장하는 것.
- 로그 파일은 그렇게 저장한 파일을 의미한다.
- 비행기로 치자면 블랙 박스라고 할 수 있다.
- 비행기 사고가 났을 때 블랙 박스를 찾아 분석하는 것처럼, 시스템 로그는 시스템에 이상이 생기거나 해킹을 당했을 때 분석하는 용도로 쓰인다.
- `syslogd` 데몬이 부팅 시 실행되어 각종 시스템 활동을 기록한다.

`log f + log i + log l + 1`

- 로그 파일은 기본적으로 `/var/log` 폴더에 저장된다.
- `cron` : cron 데몬 로그 파일
- `lastlog` : 각 유저의 마지막 로그인 정보. `lastlog` 명령어로 확인한다.
- `messages` : OS에서 보내지는 실시간 메세지.
- `utmp` : 시스템에 현재 로그인한 유저들의 정보. `who`, `u`, `finger` 명령어에서 사용.
- `wtmp` : 모든 로그인/로그아웃과 시스템 재부팅을 기록. `last` 명령어에서 사용.
- `dmesg` : 시스템 부팅 시 나왔던 메세지들.
- `boot` : 부트 메세지
- 한번 `cat` 명령어로 로그를 확인해보자.
- `/etc/syslog.conf` 파일의 설정에 따라 각 위치가 결정된다.

시스템 로그 관리하기

- 로그 파일이 쌓이면 용량이 늘어난다.
- 그렇다고 무작정 지우면 위기 상황 때 이전 로그를 볼 수 없게 된다.
- 그렇다면 이전 로그 파일을 다른 곳에 압축해두면 된다!
- `mv /var/log/messages /var/log/messages-backup`
- `cp /dev/null /var/log/messages`
- `gzip /var/log/messages-backup`
- 이걸 `cron`으로 처리하면 완벽할 것 같지만...
  이렇게 하면 이전 백업 파일은 덮어씌어져버린다!

Log rotation

- 백업을 여러개 만들어놓고, 일정 수(10개쯤) 이상이 되면 오래된 것 부터 덮어쓴다.
- `savelog`, `logrotate` 같은 쉘 스크립트가 미리 작성되어 있다.
- 스팍스 서버에서는 `logrotate`를 쓰고 있다.
- `ls /var/log`로 직접 확인해 보자.

```
1 #!/bin/sh
2
3 # Clean non existent log file entries from status file
4 cd /var/lib/logrotate
5 test -e status || touch status
6 head -1 status > status.clean
7 sed 's/"//g' status | while read logfile date
8 do
9     [ -e "$logfile" ] && echo "\"$logfile\" $date"
10 done >> status.clean
11 mv status.clean status
12
13 test -x /usr/sbin/logrotate || exit 0
14 /usr/sbin/logrotate /etc/logrotate.conf
~
                                        1,1          모두
```

- `/etc/logrotate.conf` : 설정 파일

logrotate.conf

```
# see "man logrotate" for details
# rotate log files weekly
weekly

# keep 4 weeks worth of backlogs
rotate 4

# create new (empty) log files after rotating old ones
create

# uncomment this if you want your log files compressed
#compress

# packages drop log rotation information into this directory
include /etc/logrotate.d

# no packages own wtmp, or btmp -- we'll rotate them here
/var/log/wtmp {
    missingok
    monthly
    create 0664 root utmp
    rotate 1
}

/var/log/btmp {
    missingok
    monthly
    create 0660 root utmp
    rotate 1
}

# system-specific logs may be configured here
                                                    1,1          모두 1
```

- `/etc/logrotate.d` : 각종 프로그램의 로그의 로테이션 관련 설정을 담는다.

logrotate.conf 2

```
/var/log/apache2/*.log {
    weekly
    missingok
    rotate 52
    compress
    delaycompress
    notifempty
    create 640 root adm
    sharedscripts
    postrotate
        /etc/init.d/apache2 reload > /dev/null
    endscript
}
                                    1,1          모두
```

- `delaycompress` : 압축을 다음 로테이션까지 미룬다.
- `notifempty` : 비어있을 경우에는 로테이트를 하지 않는다.
- `postrotate` ~ `endscript` : 사이에 있는 스크립트를 로테이션 이후 실행한다.
- `sharedscripts` : 각 파일별로 실행되는 스크립트를 한번만 실행시킨다.
- `mail 메일주소` : 해당 메일 주소로 덮어 씌워진 로그 파일을 메일로 보낸다.
- `size 크기` : 로테이션 뒤의 파일 사이즈가 이 크기를 넘지 않도록 한다.
  100k, 10M 등으로 크기를 지정할 수 있다.

`logcheck`

- 로그 파일은 보통 길고 쓸데없는 내용을 많이 담는 것 같다.
- `logcheck` : 시스템 로그를 분석해서 설정한 사용자에게 메일로 알려주는 쉘 스크립트이다.
- 매번 시스템에 접속하지 않고, 로그를 일일이 읽지 않아도 보안 위반사항이나 심각한 에러 등을 확인할 수 있다.
- 부팅 시, 그리고 매 시각마다 cron에서 실행한다. (`cat /etc/cron.d/logcheck`)

`logcheck` 설정

- `/etc/logcheck`에는 각종 설정 파일들과 폴더들이 들어있다.
- `logcheck.conf`로 REPORTLEVEL과 메일을 보낼 곳 등을 설정할 수 있다.
- `logcheck.logfiles`에는 검사할 로그파일들의 목록이 들어있다.
- 이 로그파일들은 `egrep`으로 필터링한 뒤 메일로 보내준다.
- 필터링은 세 단계로 이루어진다.
- SECURITY ALERTS : 침입 시도 흔적을 검사. `cracking.d`와 `cracking.ignore.d`로 필터링.
- SECURITY EVENTS : 덜 치명적이지만 주의가 필요한 사건. `violations.d`와 `violations.ignore.d`로 필터링.
- SYSTEM EVENTS : 나머지 메세지들. 무시된 결과는 포함하지 않는다.

REPORTLEVEL

- SYSTEM EVENT를 필터링하는 강도.
- `ignore.d.paranoid`, `ignore.d.server`, `ignore.d.workstation`
- paranoid : ignore하는 패턴이 제일 적다.
  소수의 서비스를 제공하는, 높은 보안 수준을 가질 시스템에 적합하다.
- server : 기본적인 필터링 레벨. 여러 데몬에 대한 ignore 패턴이 있다.
- workstation : 안전한 시스템에 사용. 대부분의 메세지를 필터링한다.

fail2ban

- `/var/log/pwdfail`, `/var/log/apache/error_log`같은 로그를 읽어, 패스워드를 너무 많이 틀리는 IP를 차단한다.
- `filter.d` : 로그인 실패 등에 해당하는 정규 표현식들이 있는 디렉토리.
- `action.d` : 상황에 따라 실행할 여러 명령어들이 들어있는 디렉토리.
- `jail.conf` : 각 필터와 액션을 이어주는 설정 파일.

fail2ban.org에서 자세한 도움말을 볼 수 있다.

참고 자료

- http://wikipedia.org
- http://www.unixgeeks.org/security/newbie/unix/cron-1.html
- http://www.fis.unipr.it/pub/linux/redhat/9/en/doc/RH-DOCS/rhl-cg-ko-9/s1-autotasks-anacron.html
- http://www.superuser.co.kr/linux/logrotate/page04.htm
- 작년 세미나 자료