시스템프로그래밍 (01분반)

.bashrc해석 및 prompt변경

교수 - 장경식 교수님
학번 - 2011136099
성명 - 임용민
제출일 - 2016년 10월 7일

1. .bashrc의 해석

1 - PART 1

```
# ~/.bashrc: executed by bash(1) for non-login shells.
# see /usr/share/doc/bash/examples/startup-files (in the package
bash-doc)
# for examples

# If not running interactively, don't do anything
case $- in
    *i*) ;;
      *) return;;
esac
```

- **$-** : 현재 셸 옵션을 저장하는 환경변수다. 옵션의 값은 아래와 같다.

  h: hash all, i: interactive, m: monitor, B: brace expand, H: hist [history] expand, e: errexit, C: noclobber

- 위 case 문은 $- 변수값(문자열) 중 문자 i가 포함되었는지 확인한다. i 값은 현재 셸의 interactive 여부를 뜻한다. 그리고 포함이 되어있다면 다음 스크립트를 실행하고 그렇지 않다면 현재 실행중인 .bashrc 스크립트를 종료한다.
- 즉, 현재 셸이 interactive 하지 못하면 스크립트를 종료한다.

1 - PART2

```
# don't put duplicate lines or lines starting with space in the
history.
# See bash(1) for more options
HISTCONTROL=ignoreboth
```

- **HISTCONTROL** 변수를 ignoreboth 값으로 설정한다. HISTCONTROL 변수는 중복되는 명령에 대한 기록 여부를 저장하는 변수다. HISTCONTROL 변수가 가질 수 있는 값은 다음과 같다.

  1. ignorespace: 스페이스 공백으로 시작되는 명령은 무시한다.
  2. ignoredups: 중복되는 명령은 무시한다.
  3. ignoreboth: 위 1, 2 두 경우 모두 무시한다.

- HISTCONTROL 변수를 ignoreboth 값으로 설정했으므로 명령어 **history**에서 스페이스 공백으로 시작되었던 명령어와 중복된 명령어는 찾을 수 없다.

1 - PART3

```
# append to the history file, don't overwrite it
shopt -s histappend

# for setting history length see HISTSIZE and HISTFILESIZE in bash(1)
HISTSIZE=1000
HISTFILESIZE=2000

# check the window size after each command and, if necessary,
# update the values of LINES and COLUMNS.
shopt -s checkwinsize

# If set, the pattern "**" used in a pathname expansion context will
# match all files and zero or more directories and subdirectories.
#shopt -s globstar
```

- **shopt**는 shell option의 약자다. 명령어 *shopt*를 사용해서 현재 옵션 전체를 조회할 수 있다. **-s**옵션은 뒤에 나올 설정을 set한다는 것이다. 즉 위 코드에서는 셸 옵션 중 histappend 설정을 set한다는 것이다. 만일 histappend 옵션이 unset되어 있다면 history file에 새로운 항목을 덧붙여 추가하는 것이 아니라, 이전 내용에 덮어쓴다.
- **HISTSIZE**는 BASH 세션이 진행되는 동안 메모리에 저장되는 history list의 명령어 수다. HISTSIZE를 1000으로 설정하였다.
- **HISTFILESIZE**는 BASH 세션을 시작하거나, 끝날때 참조하는 history file의 명령어 항목 수다. HISTFILESIZE를 2000으로 설정하였다.
- shopt 명령어를 통해 checkwinsize 옵션을 set하였다. 이는 Bash가 각 명령어가 끝났을 때 필요하면 터미널 windows의 LINES와 COLUMNS를 받아와 업데이트 할 수 있다는 옵션이다.
- shopt 명령어를 통해 globstar를 set하게되면 경로에서 "**"을 사용하여 확장 글로빙을 할 수 있다. 이를 사용하면 모든 파일과 0개 이상의 디렉토리과 그 하위 디렉토리와 매칭된다.

1 - PART4

```
# make less more friendly for non-text input files, see lesspipe(1)
[ -x /usr/bin/lesspipe ] && eval "$(SHELL=/bin/sh lesspipe)"
```

- **[** 은 test와 같은 기능을 하는 명령어다. 가독성을 위해 [명령을 사용하는 경우에]를 덧붙인다. [의 -x 옵션은 /usr/bin/lesspipe가 실행가능한 파일이면 참을 반환한다.
- **&&**은 AND 목록이다. AND 목록은 일련의 명령들을 수행할 때 사용한다. 한 명령이 참이면 차례로 다음 명령을 수행하고 거짓인 명령이 있으면 그 즉시 수행을 끝낸다.
- **eval "$(SHELL=/bin/sh lesspipe)"**
  - **eval**: 주어진 인수들을 평가(evaluation)한다.
  - $( ): 어떤 명령의 수행결과를 갈무리할 때 사용한다. $(명령) 구문의 결과는 문자열 출력이다.
  - " "(큰따움표) 쌍 안에 변수 표현 식이 있으면 해당 줄이 실행 될 때 변수 표현 식이 해당 변수의 내용으로 치환된다.
  - 위 코드를 순서대로 해석하면 다음과 같다.
    1. $ SHELL=/bin/sh lesspipe (엔터)
       export LESSOPEN="| /usr/bin/lesspipe %s";
       export LESSCLOSE="/usr/bin/lesspipe %s %s";
    2. $ echo "$(SHELL=/bin/sh lesspipe)" (엔터)
       export LESSOPEN="| /usr/bin/lesspipe %s";
       export LESSCLOSE="/usr/bin/lesspipe %s %s";
    3. eval "$(SHELL=/bin/sh lesspipe)"
       - 즉 위 코드에서 eval은 갈무리된 명령결과로 반환된 위 두 개의 명령(export로 시작하는)을 실행한다. LESSOPEN과 LESSCLOSE 환경변수를 설정한다.

1 - PART5

```
# set variable identifying the chroot you work in (used in the prompt
below)
if [ -z "${debian_chroot:-}" ] && [ -r /etc/debian_chroot ]; then
    debian_chroot=$(cat /etc/debian_chroot)
fi
```

- **chroot**란 change root의 줄임말이다. system jail이라고도 표현한다. Root 계정을 제외한 계정에서 home 디렉터리를 루트 계정인 것마냥 인식하게 만들어 시스템과 관련된 디렉터리로 접속을 차단하여 보안을 강화한다.
- [의 -z 옵션은 뒤에 나오는 string(문자열)이 널이면 참을 반환한다.
- **${debian_chroot:-}**은 매개변수 치환 구문이다. debian_chroot가 널이면 **-**뒤에 나오는 값으로 치환한다.
- [의 -r 옵션은 file이 읽을 수 있으면 참을 반환한다. 즉 /etc/debian_chroot 파일이 읽을 수 있는 파일이면 참을 반환한다.
- 즉 debian_chroot 변수가 널이고 /etc/debian_chroot 파일을 읽을 수 있다면 debian_chroot의 값을 /etc/debian_chroot 파일 내용으로 설정한다.

1 - PART6

```
# set a fancy prompt (non-color, unless we know we "want" color)
case "$TERM" in
    xterm-color|*-256color) color_prompt=yes;;
esac
```

- 환경변수 **$TERM**은 터미널의 종류를 저장한다.
- '*'은 와일드카드 확장으로 임의의 문자열에 부합한다.
- 위 case 문은 환경변수 $TERM의 값이 xterm-color 이거나 -256color로 끝나면 color_prompt 변수를 yes로 설정한다.

1 - PART7

```
# uncomment for a colored prompt, if the terminal has the capability;
turned
# off by default to not distract the user: the focus in a terminal
window
# should be on the output of commands, not on the prompt
#force_color_prompt=yes

if [ -n "$force_color_prompt" ]; then
    if [ -x /usr/bin/tput ] && tput setaf 1 >&/dev/null; then
      # We have color support; assume it's compliant with Ecma-48
      # (ISO/IEC-6429). (Lack of such support is extremely rare, and
such
      # a case would tend to support setf rather than setaf.)
      color_prompt=yes
    else
      color_prompt=
    fi
fi
```

- [의 -n 옵션은 뒤에 나오는 String이 널이 아니면 참을 반환한다.
- **/dev/null**은 범용 "비트 휴지통"이다.
- 위 코드는 force_color_prompt가 널이 아니고 /usr/bin/tput이 실행가능한 파일이면 tput setaf 1 명령의 출력 내용을 범용 비트 휴지통에 모두 버린 후 color_prompt 변수를 yes 로 설정한다. 만일 /usr/bin/tput이 실행가능하지 않거나 존재하지 않다면 color_prompt 의 변수를 비워둔다.

1 - PART8

```
if [ "$color_prompt" = yes ]; then
    PS1='${debian_chroot:+($debian_chroot)}\[\033[01;32m\]\u@\h\
[\033[00m\]:\[\033[01;34m\]\w\[\033[00m\]\$ '
else
    PS1='${debian_chroot:+($debian_chroot)}\u@\h:\w\$ '
fi
unset color_prompt force_color_prompt
```

- 변수 color_prompt의 값이 yes면 1차 명령 프롬프트 변수인 **PS1**을 위 코드에 나온 형식으로 지정해준다.

  \[은 비출력문 시퀀스 시작을 의미하고, \]은 비출력문 시퀀스 끝을 의미한다.

- 변수 color_prompt의 값이 yes가 아니면 **PS1**을 else 이후에 나온 형식으로 지정해준다.

  PS1='${debian_chroot:+($debian_chroot)}\u@\h:\w\$ '

  \u: 현재 사용자의 사용자명

  \h: 호스트명

  \w: 현재 작업디렉터리의 완전경로

- **unset** 명령은 현재 셸 환경에서 변수나 함수를 제거한다. 즉, color_prompt 변수와 force_color_prompt 변수를 제거한다.

1 - PART9

```
# If this is an xterm set the title to user@host:dir
case "$TERM" in
xterm*|rxvt*)
    PS1="\[\e]0;${debian_chroot:+($debian_chroot)}\u@\h: \w\a\]$PS1"
    ;;
*)
    ;;
esac
```

- 위 case 문은 환경변수 $TERM의 값이 xterm으로 시작하거나 rxvt로 시작하면 $PS1 변수를 다음과 같이 설정하라는 것이다.
- 그 이외의 경우에는 아무일도 하지 않는다.

  '*)': 와일드카드를 사용하여 그 외의 모든 경우를 나타내고 있다.

1 - PART10

```
# enable color support of ls and also add handy aliases
if [ -x /usr/bin/dircolors ]; then
    test -r ~/.dircolors && eval "$(dircolors -b ~/.dircolors)" ||
eval "$(dircolors -b)"
    alias ls='ls --color=auto'
    #alias dir='dir --color=auto'
    #alias vdir='vdir --color=auto'

    alias grep='grep --color=auto'
    alias fgrep='fgrep --color=auto'
    alias egrep='egrep --color=auto'
fi

# colored GCC warnings and errors
#export
GCC_COLORS='error=01;31:warning=01;35:note=01;36:caret=01;32:locus=01
:quote=01'
```

- ls와 grep등의 color 지원을 alias명령어를 통해 설정하고 있다.
- **alias**는 자주 사용하는 명령어를 간편하게 사용하기 위해 특정문자로 설정해주는 명령어다.
- 위 if문을 보면 /usr/bin/dircolors가 실행 가능한 파일이면 다음을 수행한다.

    1. ~/.dircolors 파일이 읽을 수 있는지 확인한다. 읽을 수 있다면 eval 명령어를 통해서 dircolors -b ~/.dircolors의 결과를 실행한다. ~/.dircolors 파일이 읽을 수 없거나 존재하지 않으면 dircolors -b의 결과를 실행한다.
    2. alias 명령어를 이용해 ls, dir, vdir, grep, fgrep, egrep의 color 지원을 활성화한다.

- GCC의 warnings과 errors에 대해 color지원을 하고 싶다면 그 아래의 내용을 주석처리하지 않는다.

    export GCC_COLORS='error=01;31:warning=01;35:note=01;36:caret=01;32:locus=01:quote=01'

1 - PART11

```
# some more ls aliases
alias ll='ls -alF'
alias la='ls -A'
alias l='ls -CF'
```

- alias 명령어를 통해 ls명령어에 관한 추가적인 간편 설정을 한다.

1 - PART12

```
# Add an "alert" alias for long running commands.  Use like so:
#   sleep 10; alert
alias alert='notify-send --urgency=low -i "$([ $? = 0 ] && echo
terminal || echo error)" "$(history|tail -n1|sed -e '\''s/^\s*[0-9]\+
\s*//;s/[;&|]\s*alert$//'\'')"'
```

1. --urgency=low
   - -u, --urgency는 긴급함의 정도를 나타낸다 (low, normal, critical)
2. -i "$([$? = 0] && echo terminal || echo error)"
   - -i, --icon=ICON[,ICON…]: 화면에 표시할 아이콘 파일을 지정한다.
   - **$?**은 최근 실행시킨 프로그램의 종료 값을 저장하는 환경변수다.
   - 즉, exit code가 0이면 terminal을 출력하고 그렇지 않으면 error를 출력한다.
2. $(history|tail -n1|sed -e '\"s/^\s*[0-9]+\s*//;s/[;&|]\s*alert$//'\")
   - history|tail -n1은 history에 있는 마지막 커맨드를 반환한다.
   - sed -e '\"s/^\s*[0-9]+\s*//;s/[;&|]\s*alert$//'\")는 두 개의파트로 구분할 수 있다.

     3.1 **sed 's/^\s*[0-9]+\s*//'**: 앞에 나오는 공백과 탭을 제거하고 모든 숫자를 제거한다. 그리고 뒤에 나오는 공백과 탭을 제거한다.

     3.2 **s/[;&|]\s*alert$//**: 앞에 나오는 ';', '&', '|', 탭과 공백을 제거한다. 단어 "alert"도 제거한다.
   - 이는 마지막 실행한 명령어를 깨끗하게 만들기 위한 것이다.

1 - PART13

```
# Alias definitions.
# You may want to put all your additions into a separate file like
# ~/.bash_aliases, instead of adding them here directly.
# See /usr/share/doc/bash-doc/examples in the bash-doc package.

if [ -f ~/.bash_aliases ]; then
    . ~/.bash_aliases
fi
```

- [의 -f 옵션은 file이 정규적인 파일이면 참이다.
- '.(마침표)명령'은 현재 셸에서 명령을 수행하게 한다. 일반적으로 스크립트에서 외부 명령이나 외부 스크립트를 수행하는 경우 하위 셸에서 실행된다. 그러나 마침표 명령을 사용하면 호출한 스크립트와 같은 셸 안에서 수행된다.
- 즉, ~/.bash_aliases가 정규적인 파일이면 현재 셸에서 ~/.bash_aliases를 실행한다.

1 - PART14

```
# enable programmable completion features (you don't need to enable
# this, if it's already enabled in /etc/bash.bashrc and /etc/profile
# sources /etc/bash.bashrc).
if ! shopt -oq posix; then
  if [ -f /usr/share/bash-completion/bash_completion ]; then
    . /usr/share/bash-completion/bash_completion
  elif [ -f /etc/bash_completion ]; then
    . /etc/bash_completion
  fi
fi
```

- shopt 명령어의 -o 옵션은 뒤에오는 optname의 설정 값을 확인할 수 있다. -q 옵션은 출력하지 않는다.
- 즉, shopt의 posix가 off라면 다음을 시행한다.
  - /usr/share/bash-completion/bash_completion이 정규적인 파일이면 현재 셸에서 /usr/share/bash-completion/bash_completion을 실행한다. 만일 그렇지 않고 /etc/bash_completion가 정규적인 파일이면 현재 셸에서 /etc/bash_completion을 실행한다.

2. 나만의 Prompt 꾸미기

2-1 .bash_profile

```
export PS1="\[\033[0;36m\][\[\033[0;37m\]\d-\t-\[\033[0;34m\]\u@\\
[\033[0;36m\]\W]\$ "
#export PS1='${debian_chroot:+($debian_chroot)}\[\033[01;32m\]\u@\h\
[\033[00m\]:\[\033[01;34m\]\w\[\033[00m\]\$ '
```

- **$PS1**: 1차 명령 프롬프트 변수
  - **PS1**의 값을 적절하게 바꾸면 원하는 형식의 프롬프트로 설정 할 수 있다.
- **export**
  - export 명령은 변수 이름을 하위셸의 매개변수들로 사용할 수 있게 만든다. 기본적으로 한 셸안에서 생성된 변수는 그 셸에서 호출한 하위셸들에서 사용하지 못한다. export 명령으로 내보낸 변수는 원래 셸에서 파생된 임의의 자식 프로세스들에서 환경 변수로 존재하게 된다.
- **.bash_profile**
  - 셸 로그인 시 적용되는 사용자별 셸 환경에 대한 설정 파일이다.
  - 이들 환경 변수들은 오직 그 사용자에게만 한정되며, 그 이외의 다른 사람에게는 영향을 미치지 않는다. 이 파일은 전역적인 설정 파일인 /etc/profile이 실행된 후 실행된다.
  - 일반적으로 PS1은 /etc/profile에 정의되는데, ~/.bash_profile에 다시 정의함으로써 해당 사용자의 프롬프트 설정만 재정의 할 수 있다.
  - 오직 root 권한을 가진 자만이 /etc/profile의 내용을 변경할 수 있으므로, 각 사용자는 ~/.bash_profile에서 환경 변수를 재정의함으로써 자신만의 환경을 설정한다.

- PS1 변수에 사용되는 기호

variable	note
\a	an ASCII bell character (07)
\d	the date in "Weekday Month Date" format (e.g., "Tue May 26")
\D{format}	the format is passed to strftime(3) and the result is inserted into the prompt string; an empty format results in a locale-specific time representation. The braces are required
\e	an ASCII escape character (033)
\h	the hostname up to the first '.'
\H	the hostname
\j	the number of jobs currently managed by the shell
\l	the basename of the shell's terminal device name
\n	newline
\r	carriage return
\s	the name of the shell, the basename of $0 (the portion following the final slash)
\t	the current time in 24-hour HH:MM:SS format
\T	the current time in 12-hour HH:MM:SS format
\@	the current time in 12-hour am/pm format
\A	the current time in 24-hour HH:MM format
\u	the username of the current user
\v	the version of bash (e.g., 2.00)
\V	the release of bash, version + patch level (e.g., 2.00.0)
\w	the current working directory, with $HOME abbreviated with a tilde (uses the $PROMPT_DIRTRIM variable)
\W	the basename of the current working directory, with $HOME abbreviated with a tilde
\!	the history number of this command
\#	the command number of this command
\$	if the effective UID is 0, a #, otherwise a $
\nnn	the character corresponding to the octal number nnn
\\	a backslash
\[	begin a sequence of non-printing characters, which could be used to embed a terminal control sequence into the prompt
\]	end a sequence of non-printing characters

- **ANSI color**
  - ANSI 컬러는 ANSI escape code의 기능 중 하나이다. ANSI escape code는 터미널의 텍스트 포맷을 제어하기 위해서 만들어진 코드이며 ISO/IEC-6429 표준으로 제정되어있다.
  - ANSI Escape sequence = CSI + n [;+ ...] + letter
    - CSI(Control Sequence Introducer): ANSI escape sequence에서 시작문자

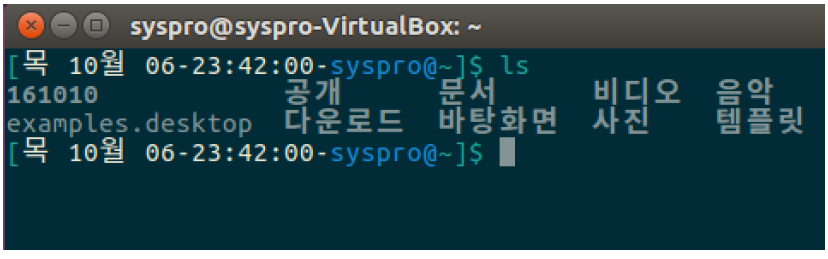
      CSI = \e[= \033[
    - n [;+ …]: n은 숫자다. 복수개가 나올때는 세미콜론(;)으로 구분한다
    - letter: 영문자 1개, ANSI escape sequence의 명령(command)에 해당한다.

      T: 스크롤 다운, m: 색상 변경 등
  - 색상표

Intensity	0	1	2	3	4	5	6	7	9
Normal		Red	Green	Yellow	Blue	Magenta	Cyan	White	reset
Bright	Black	Red	Green	Yellow	Blue	Magenta	Cyan	White	

  - 예제
    - \033[0;36m : \033[는 CSI다. 그 후 나오는 0은 Reset / Normal 기능으로 앞서 설정된 속성들을 초기화한다. 두 번째 인수인 36에서 일의 자리인 6은 Cyan 색상을 나타낸다. 십의 자리인 3은 Cyan 색 중 normal intensity를 선택해 글자색으로 변경하라는 뜻이다. m은 색상 변경 명령이다. 즉 해당 명령어 이후의 글자는 Cyan 색으로 나타난다.
- 나의 프롬프트
  - ```
    export PS1="\[\033[0;36m\][\[\033[0;37m\]\d-\t-\[\033[0;34m\]\u@\\[\033[0;36m\]\W]\$ "
    ```
  - 해석

    (Cyan) "[" (White) "현재 날짜-현재시간" (Blue) "-현재사용자명"@ (Cyan) "현재 디렉토리의 basename$ "
  - 결과


    ```

3. 고찰 및 느낀점

- .bashrc의 내용을 해석하다 보니 자연스럽게 셸 스크립트를 공부하기 위해 책과 학습 자료를 펼쳤습니다. 셸 스크립트에 대한 이해 없이 무작정 내용을 해석하려고 하니 애매한 부분과 이해가 안 가는 곳이 대다수였기 때문입니다. 예를 들어 부울 판정 구문인 '['에 대해서 학습을 하기 이전에는 단순 괄호로 생각했습니다. 그러나 학습을 통해 '['이 test 명령과 동의어라는 것을 알고 나니 셸 스크립트의 이해도가 높아졌습니다. '[' 뿐만 아니라 따옴표 처리와 같은 세세한 규칙들부터 'AND 목록', 'OR 목록'을 이용한 일련의 명령어 처리, 'eval', '.', 등과 같은 명령들 그리고 정규표현식과 매개변수 치환 구문 등 다양한 셸 스크립트의 문법을 공부하면서 리눅스 셸의 강력함이 무엇인지 와 닿았습니다. 간단하고 축약된 기능들을 사용해 강력한 기능을 더욱 쉽게 구현할 수 있는 셸 스크립트의 매력을 느끼게 되어 앞으로 개발하게 될 프로그램의 프로토타입 제작용으로 요긴하게 쓰게 될 것 같습니다. 또한, 나만의 프롬프트를 만들고 터미널을 사용하니 같은 리눅스지만 다른 시스템에 설치된 리눅스보다 제 컴퓨터에 설치된 리눅스에 더욱 애정이 생겼습니다.