**System Programming**

1분반(Lab-1)



작 성 자 : 김형근

학 번 : 2011136033

담 당 교 수 : 장경식 교수님

제 출 일 : 2016. 10. 07

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

로그인되어 있지 않은 쉘에 대해서 bashrc내용이 실행된다. 하지만, 실제로 로그인된 쉘에 대해서도 실행이 된다. 이에 대해서 .profile내용을 확인하면

|  |
| --- |
| # ~/.profile: executed by the command interpreter for login shells.  # This file is not read by bash(1), if ~/.bash\_profile or ~/.bash\_login  # exists.  # see /usr/share/doc/bash/examples/startup-files for examples.  # the files are located in the bash-doc package.  # the default umask is set in /etc/profile; for setting the umask  # for ssh logins, install and configure the libpam-umask package/  # umask 022  # if running bash  If [ -n “$BASH\_VERSION” ]; then  # include .bashrc if it exists  If [ -f “$HOME/.bashrc” ]; then  . “$HOME/.bashrc”  Fi  Fi  #set PATH so it includes user’s private bin directories  PATH=”$HOME/bin:$HOME/.local/bin:$PATH” |

로 되어있다. 여기에서 보면 .bash\_profile이나 .bash\_login파일이 없을 경우 .profile파일이 로그인된 쉘에 의해서 실행되며, .profile파일은 bash가 실행되어 있을 경우 홈디렉토리에 .bashrc파일이 존재하면 .bashrc파일을 참조한다.

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

상호작용 쉘이 아닐 경우, 어떤 것도 하지않는다.

여기서 상호작용 쉘이란 -s옵션으로 지정하지 않으면, -c옵션으로 지정하는 것 없이 비옵션 인수 없이 시작되고 입력과 에러 출력이 모두 -i 옵션에 시작되거나 터미널에 연결되어 있는 쉘을 말한다. 상호작용 쉘은 일반적으로 유저 터미널에서 읽히고 쓰인다.

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

HISTCONTROL은 어떻게 명령어들을 히스토리상에 저장하는지를 제어하는지에 대한 값이다. 몇몇값들에 대해서 소개하면 ignorespace 빈 공간으로 시작하는 명령어를 히스토리에 저장하지 않는다. 그리고 ignoredups는 이전 값과 같은 명령어를 저장하지 않는다. Ignoreboth는 ignorespac와 ignoredups를 합쳐놓은 값이다. 그리고 erasedups는 모든 이전 라인과 현재 라인을 대조하여 같은 명령어를 히스토리에서 제거한다.

|  |
| --- |
| #append to the history file, don't overwrite it  shopt -s histappend  #for setting history length see HISTSIZE and HISTFILESIZE in bash(1) |

Shopt 설정문은 추가적인 쉘의 기능적 행동을 변경을 가능하게 한다. 여기에서 -s옵션은 해당 옵션명을 사용 가능하게 한다.

Histappend 옵션은 설정할 경우 히스토리 리스트가 HISTFILE변수에 설정된 파일에 추가된다.

HISTSIZE와 HISTFILESIZE는 각각 히스토리 크기에 대해서 설정하며 HISTFILESIZE는 히스토리 파일의 크기를 설정한다. HISTSIZE가 음수일 경우에는 모든 명령어들을 히스토리에 기록한다.HISTFILESIZE의 기본 값은 HISTFILESIZE로 되어있다.

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

각 명령어 이후에 윈도우의 크기를 확인하고, 필요하면 라인(행)과 열을 갱신한다.

|  |
| --- |
| # if set, the pattern “\*\*” used in a pathname expansion context will  # match all files and zero or more directories and subdirectories.  # shopt -s globstar스 이름 # \n: newline |

globstar옵션은 \*\*을 사용하면 모든 파일과 0 또는 좀더 많은 디렉토리와 서브디렉토리를 대조한다. '/' 다음에 사용하는 경우 오직 디렉토리와 서브디렉토리만 대조한다.

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

less를 비텍스트 파일에 좀더 친숙하게 만든다. 여기에서 쓰이는 방법은 lesspipe를 사용하는 방법이며, lesspipe는 lesspipe와 lessfile을 합쳐놓은 프로그램이다. lesspipe는 표준출력으로 less프로그램에 문맥/정보를 입력한다. 이는 lesspipe는 유저에게 less 프로그램이 디코딩하도록 기리다리는 시간을 줄여준다. Lessfile은 lesspipe에 의해 생성된 less에 읽히고 필요없는 파일들을 삭제한다.

|  |
| --- |
| #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 |

-z 옵션은 만약 뒤에 나온 문자열이 null일 경우 True값을 반환한다. -r는 뒤에 파일이 readable일 경우 True값을 반환한다. 따라서, ${debian\_chroot:-}가 null값이고 /etc/debian\_chroot를 읽을 수 있을 경우 debian\_chroot에 cat /etc/debian\_chroot값을 대입한다.

|  |
| --- |
| # set a fancy prompt (non-color, unless we know we “want” color)  case “$TERM” in  xterm-color) color\_prompt=yes;;  esac |

팬시한 프롬프트를 설정한다. $TERM의 값이 xterm-color일 경우 색상이 입혀진다.

|  |
| --- |
| # uncomment for a colored prompt, if the terminal has capability; turned  # off by default to not distract the us스 이름 # \n: newlineer: the focus in a terminal window  # 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 extreamly rare, and such  # a case would tend to support setf rather than setaf.  color\_prompt=yes  else  color\_prompt=  fi  fi |

터미널이 성능이 된다면 색이 입혀진 주석을 사용한다.; 유저의 정신을 혼란하게 만들지 않기 위해 Ubuntu에서는 기본값이 해제되어있다.

force\_color\_prompt가 yes값으로 되어있다면, Ecma-48(ISO/IEC-6429)의 색상과 호환이 되는 경우 color\_prompt값이 yes가 되고 호환이 되지 않거나 force\_color\_prompt값이 yes가 아닐경우 color\_prompt값는 빈 값이 된다.

|  |
| --- |
| 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  # if this is an xterm set the title to [user@host](mailto:user@host):dir  case “$TERM” in  xterm\*|rxvt\*)  PS1=”\[\e]0;${debian\_chroot:+($debian\_chroot)}\u@\h:\w\a]$PS1”  ;;  \*)  ;;  esac |

위에서 color\_prompt가 yes로 설정되었다면, 색상이 입혀진 prompt로 설정되고 아닐 경우에는 색상이 없는 프롬프트로 설정된다.

색상을 적용하는 방법은 \[\033[컬러넘버m\]를 해당 프롬프트 제어 문자 앞에 설정하면 되며, \[\033[00m\]으로 하면 기본값으로 돌아간다.

컬러넘버 몇가지는 다음과 같다.

Black 0;30 Dark Gray 1;30

Blue 0;34 Light Blue 1;34

Green 0;32 Light Green 1;32

Cyan 0;36 Light Cyan 1;36

Red 0;31 Light Red 1;31

Purple 0;35 Light Purple 1;35

Brown 0;33 Yellow 1;33

Light Gray 0;37 White 1;37

그리고 출력되는 프롬프트 제어문자는 PS1의 값 그대로인 유저이름(\u)@호스트이름(\h):현재경로(\w)\$가 된다. 그외의 제어문자는 다음과 같다

\t: 24시간 기준 시:분:초

\T: 12시간 기준 시:분:초

\@: 오전/오후

\a: ASCII벨 문자(07)

\d: 요일 달 일(예시: Tue May 26)

\e: ASCII 이스케이프 문자

\H: 호스트 전체 이름

\j: 현재 쉘에서 관리되는 작업의 수

\n: 개행

\v: 배쉬 버전(2.0)

\V: 배포 배쉬, 버전+패치레벨(예시: 2.0.1)

\W: 현재 작업중인 디렉토리의 이름

\!: 히스토리 넘버

\#: 커맨드 넘버

\[: 프롬프트에서 터미널 제어 시퀸스에 사용되는 비출력 문자의 시작지점

\]: 비출력 문자의 끝

그리고 unset을 통해 color\_prompt와 force\_color\_prompt값을 해제한다.

터미널의 타이틀을 유저이름@호스트이름:현재 작업 디렉토리이름으로 한다.

|  |
| --- |
| # 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  # some more ls aliases  alias ll='ls -alF'  alias la='ls -A'  alias l='ls -CF'  # 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$//'\'')”'  # 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 |

디렉토리 색상을 실행할 수 있다면, 즉 /usr/bin/dircolors를 실행가능하면 ls 및 다른 몇몇 명령어들에 색상을 덮어 씌우고 몇몇 명령어를 추가한다.

여기에서 보면 “alias 추가할명령어='명령어의 내용'”을 추가하면 실제로 해당 명령어를 사용할 수 있다.

bashrc에서 장기적으로 동작하는 명령어를 위하여 alert 명령어를 추가한다. 사용은 다음과 같다: sleep 10; alert

명령어 정의에서 추가할 명령어는 Ubuntu에서는 .bash\_aliases파일같은 분산된 파일에 추가하는 것을 권장한다.

|  |
| --- |
| #enable programmable completion features (you don't need to enable  # this, if it's already enabled in /etc/bash.bashrc and /etc/profile  # source /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 |

shopt의 -q 옵션은 일반 출력을 억누른다. 반환하는 값은 해당 optname이 설정되었는 해제되어있는가를 가르킨다. 인수가 전부 활성화 되어있으면 0값을 반환하고 아니면 0이 아닌 값을 반환한다. -o옵션은 optname의 값을 제한한다. 그리고 posix가 활성화 되어있으면 프로그램 가능한 bash-completion을 활성화한다.