**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-dc)  #for examples |
| # ~/.bashrc: 비 로그인 쉘을 위해 bash(1)에 의해 실행된다.  # 예제는 /usr/share/doc/bash/examples/startup-files(bash-doc 패키지 내부) 를 보아라 |
| 로그인 되어있지 않은 쉘에 대해서 bashrc 내용이 실행된다. 하지만 실제로 로그인된 쉘에 대해서도 실행이 된다. 이에 대해서는 .profile에 있다. |

실제로 ~/.profile의 내용은 다음과 같다.

|  |
| --- |
| # ~/.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 loacated 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” |
| .profile은 다음과 같다. 2번째 라인을 보면 .profile파일은 .bash\_profile이나 .bash\_login파일이 존재하면 bash(1)에 의해 읽히지 않는다.  Umask는 새폴더, 새파일의 퍼미션을 결정하는 값 또는 설정 명령어로서 /etc/umask에 설정되어 있다. 기본값은 022이다.  Bash가 실행중일 경우 .bashrc파일이 있는지 확인하고 있을 경우, login shells에서도 bashrc를 참조한다는 것을 확인할 수 있다.  그리고 가장 아래에 있는 라인은 PATH 환경변수 값이다. |

|  |
| --- |
| # if not running interactively, don’t do anything  case $- in  \*i\*);;  \*)return;;  esac |
| # 만약 상호작용하지 않는다면, 어떤것도 하지 않는다.  case $- in  \*i\*);;  \*)return;;  esac |
| 상호작용 쉘이 아닐 경우, 어떤 것도 하지 않는다. 여기에서 상호작용 쉘이란 비옵션 인자없이 시작되고 입력과 출력이 모두 –I 옵션으로 시작되거나 터미널에 연결되어 있는 쉘을 말한다. 상호작용 쉘은 일반적으로 사용자 터미널에서 읽히고 쓰인다. |

다음은 히스토리 컨트롤에 관한 내용이다.

|  |
| --- |
| # don’t put duplicate lines or lines starting with space in the history  # See bash(1) for more options  HISTCONTROL=ignoreboth  # 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 |
| # 히스토리에 중복된 라인이나 공백으로 시작하는 라인을 넣지 않는다.  # 좀더 많은 옵션에 대해서 bash(1)를 보아라.  HISTCONTROL=ignoreboth  # 히스토리 파일을 덮어쓰지 않고, 추가한다.  shopt –s histappend  # 히스토리 길이를 설정하기 위해 bash(1)에서 HISTSIZE와 HISTFILESIZE를 보아라.  HISTSIZE=1000  HISTFILESIZE=2000 |
| HISTCONTROL은 어떻게 명령어들을 히스토리상에 저장하는지를 제어하는지를 결정하는 값이다. 여러 옵션에 대해서 설명하면 ignore은 공백으로 시작하는 명령어들을 히스토리에 저장하지 않는다. 그리고 ignoredups는 이전 값과 동일한 명령어를 저장하지 않는다. ignoreboth는 ignorespace와 ignoredups를 합쳐놓은 옵션이다. erasedups는 이전의 모든 히스토리 라인과 현재 입력한 커맨드라인을 대조하여 히스토리상 같은 명령어들을 제거한다.  shopt 명령어는 추가적으로 쉘의 기능적 처리 변경을 가능하게 한다. 여기에서 –s 옵션은 사용된 옵션을 설정한다.  histappend 옵션은 설정하면 history리스트가 HISTFILE 변수로 되어있는 파일에 추가된다.  HISTSIZE와 HISTFILESIZE는 각각 히스토리 크기에 대해서 설정하며 HISTSIZE는 음수일 경우 모든 명령어들을 히스토리에 저장한다. HISTFILESIZE는 히스토리가 저장되는 파일의 크기의 값이며, 기본값은 HISTSIZE이다. |

다음은 배쉬 윈도우의 크기에 대한 내용이다.

|  |
| --- |
| # check the window size after each command and, if necessary,  # update the values of LINES and COLUMNS.  shopt –s checkwinsize |
| # 매 명령어 이후에 윈도우 크기를 확인한다. 그리고 필요하다면  # 라인(행)과 열의 크기를 갱신한다.  shopt –s checkwinsize |

다음은 globstar옵션에 대한 내용이다.

|  |
| --- |
| # if set, the pattern “\*\*” used in a pathname expansion context will  # match all files and zero or more directories and subdirectories.  # shopt –s globstar |
| # 설정한다면, 경로 확장 문맥으로 사용되는 “\*\*”패턴은  # 모든 파일과 0 또는 좀더 많은 디렉토리들과 하위 디렉토리들을 대조한다.  #shopt –s globstar |

다음은 lesspipe와 관련된 내용이다.

|  |
| --- |
| # make less more friendly for non-text input files, see lesspipe(1)  [ -x /usr/bin/lesspipe ] && eval “$(SHELL=/bin/sh lesspipe)” |
| # less를 비텍스트 입력 파일에 대하여 좀더 친숙하게 만든다, lesspipe(1)을 보아라.  [ -x /usr/bin/lesspipe] && eval “$(SHELL=/bin/sh lesspipe)” |
| Less는 more와 유사하지만, 좀더 많은 특징을 갖고 있는 프로그램이다. Less는 시작하기 전에 전체 입력 파일을 읽을 필요가 없기 때문에 큰 입력 파일을 갖고있는 것을 vi같은 텍스트 편집기보다 더 빠르다. 이러한 less를 보조하는 프로그램이 lesspipe이다. Lessfile과 lesspipe는 less에 보여지는 파일 문맥으로 변경되는데 사용된다. 이것이 의미하는 바는 less가 자동적으로 tar파일들을 열고, 압축된 파일들을 압축 풀고 심지어 그래픽 파일들에 대해서 타당한 무언가를 보여준다.  Lesspipe는 표준출력으로 문맥/정보를 전달하고 less는 넘겨받을 때마다 그것들을 읽는다. 이는 less가 처리완료된 파일을 보여주기전에 디코딩하는 것을 기다릴 필요가 없다는 것을 의미한다. |

Chroot를 식별하는 변수를 설정한다.

|  |
| --- |
| # 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) |
| -z 옵션은 뒤에 나오는 문자열이 null인 경우 True값을 반환한다. 그리고 –r는 파일이 readable할 경우 True값이 반환된다. 따라서, 두 조건을 만족하면 debian\_root에 cat /etc/debian\_chroot가 대입된다. |

Xterm에 대한 설정과 prompt에 대해 설정한다.

|  |  |
| --- | --- |
| # set a fancy prompt (non-color, unless we know we “want” color)  Case “$TERM” in  Xterm-color) color\_prompt=yes;;  Esac  # uncomment for a colord prompt, if the terminal has capability; turned  # off by default to not dstract the user: focus in terminal window  # force\_color\_prompt=yes  if [ -n “$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  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 settthe tile to user@host:dir  case “$TERM” in  xterm\*|rxvt\*)  PS1=:\[\e]0;${debian\_chroot:+($debian\_chroot)}\u@\h:\w\a]$PS1”  ;;  Else  ;;  Esac; | |
| # (우리가 우리는 색상을 원한다는 것을 아는 경우를 제외하고 비색상이다.)  case “$TERM” in  Xterm-color)color\_prompt=yes;;  Esac  # 터미널이 성능이 충분하다면, 색상입히는 것에 대해서 주석을 달지 않는다.  # ;유저가 혼란스럽지 않도록 기본값으로 꺼져있다.: 터미널 윈도우에 집중  #force\_color\_prompt=yes  if [ -n “$force\_color\_prompt” ]; then  if [-x /usr/bin/tput] && tput setaf 1 >&/dev/null; then  # 우리는 색상 지원을 갖고있다.;  # (ISO/IEC-6429)에 잘 적용된다고 가정한다.. (이러한 지원의 부족은 극적으로  # 드물고, 이러한 경우 setaf보다 setf를 지원하는 것을 더 선호한다.  color\_prompt=yes  else  color\_prompt;  fi  fi  if[ “$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  # xterm이라면 제목은 user@host:dir이다.  case “$TERM” in  xterm\*|rxvt\*)  PS1=:\[\e]0;${debian\_chroot:+($debian\_chroot)}\u@\h:\w\a]$PS1”  ;;  Else  ;;  Esac; | |
| $TERM값이 xterm-color일 경우 xterm 제목에 색상이 입혀진다.  터미널의 성능이 된다면 색이 입혀진 주석을 사용할 수 있으나, ubuntu에서는 기본값으로 제공하지 않는다.  PS1값을 수정하여 배쉬에서 사용하는 프롬프트를 커스텀마이징할 수 있다.  PS1값으로 집어넣을 수 있는 제어문자와 색상컬러는 다음과 같다.  그리고 unset으로 color\_prompt와 force\_color\_prompt변수를 반환한다. | |
| 컬러넘버(\[\033[컬러넘버m\] | |
| 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 gay 0;37 | White 1;37 |

제어문자

|  |  |
| --- | --- |
| \t | 24시간 기준 시:분:초 |
| \T | 12시간 기준 시:분:초 |
| \@ | 오전/오후 |
| \a | ASCII 벨 문자(07) |
| \d | 요일 달 일(예시: Tue May 26) |
| \e | ASCII 이스케이프 문자 |
| \h | 호스트 이름 |
| \H | 호스트 전체이름 |
| \u | 유저이름 |
| \v | 배쉬 버전 |
| \V | 배포배쉬, 버전+패치레벨 |
| \w | 현재 디렉토리 경로 |
| \W | 현재 작업중인 디렉토리의 이름 |
| \! | 히스토리 넘버 |
| \# | 커맨드 넘버 |
| \[ | 터미널 제어 시퀸스에서 사용되는 프롬프트의 비출력 문자의 시작 지점 |
| \] | 비출력 문자의 끝지점 |

명령어 추가

|  |
| --- |
| # 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. |

프로그램 가능한 completion

|  |
| --- |
| #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. |