

## .bashrc

Le fichier ~/.bashrc détermine le comportement des shells interactifs. Une étude de ce fichier peut amener une meilleure compréhension de Bash.

Etudiez le fichier avec attention, et n'hésitez pas à réutiliser certaines parties du code pour votre propre .bashrc, voire même dans vos scripts.

### Exemple de fichier .bashrc

```
#=====
#
# PERSONAL $HOME/.bashrc FILE for bash-2.05 (or later)
#
# This file is read (normally) by interactive shells only.
# Here is the place to define your aliases, functions and
# other interactive features like your prompt.
#
# This file was designed (originally) for Solaris.
# --> Modified for Linux.
# This bashrc file is a bit overcrowded - remember it is just
# just an example. Tailor it to your needs
#
#=====

# --> Comments added by HOWTO author.

#-----
# Source global definitions (if any)
#-----

if [ -f /etc/bashrc ]; then
    . /etc/bashrc # --> Read /etc/bashrc, if present.
fi

#-----
# Automatic setting of $DISPLAY (if not set already)
# This works for linux and solaris - your mileage may vary....
#-----

if [ -z "${DISPLAY:=}" ]; then
    DISPLAY=$(who am i)
    DISPLAY=${DISPLAY%%\!*}
    if [ -n "$DISPLAY" ]; then
        export DISPLAY=$DISPLAY:0.0
    else
        export DISPLAY=":0.0" # fallback
    fi
fi

#-----
# Some settings
#-----

set -o notify
set -o noclobber
set -o ignoreeof
set -o nounset
```

```

#set -o xtrace # useful for debugging

shopt -s cdspell
shopt -s cdable_vars
shopt -s checkhash
shopt -s checkwinsize
shopt -s mailwarn
shopt -s sourcepath
shopt -s no_empty_cmd_completion
shopt -s histappend histreedit
shopt -s extglob # useful for programmable completion

#-----
# Greeting, motd etc...
#-----

# Define some colors first:
red='\e[0;31m'
RED='\e[1;31m'
blue='\e[0;34m'
BLUE='\e[1;34m'
cyan='\e[0;36m'
CYAN='\e[1;36m'
NC='\e[0m' # No Color
# --> Nice. Has the same effect as using "ansi.sys" in DOS.

# Looks best on a black background....
echo -e "${CYAN}This is BASH ${RED}${BASH_VERSION%.*}${CYAN} - DISPLAY on
${RED}$DISPLAY${NC}\n"
date
if [ -x /usr/games/fortune ]; then
  /usr/games/fortune -s # makes our day a bit more fun... :-)
fi

function _exit() # function to run upon exit of shell
{
  echo -e "${RED}Hasta la vista, baby${NC}"
}
trap _exit 0

#-----
# Shell prompt
#-----

function fastprompt()
{
  unset PROMPT_COMMAND
  case $TERM in
  *term | rxvt )
    PS1="\[\h] \W > \[\033]0;[\u@\h] \w\007\" ;;
  *)
    PS1="\[\h] \W > " ;;
  esac
}

function powerprompt()
{
  _powerprompt()
  {
    LOAD=$(uptime|sed -e "s/.*: \([^,]*\).*\/\1/" -e "s/ //g")
    TIME=$(date +%H:%M)
  }
}

```

```

}

PROMPT_COMMAND=_powerprompt
case $TERM in
*term | rxvt )
PS1="{cyan}[\$TIME \$LOAD]\$NC\n[\h \#] \W > \[\033]0;[\u@\h] \w\007\" ;;
linux )
PS1="{cyan}[\$TIME - \$LOAD]\$NC\n[\h \#] \w > \" ;;
* )
PS1="[\$TIME - \$LOAD]\n[\h \#] \w > \" ;;
esac
}

powerprompt # this is the default prompt - might be slow
# If too slow, use fastprompt instead....

#=====
#
# ALIASES AND FUNCTIONS
#
# Arguably, some functions defined here are quite big
# (ie 'lowercase') but my workstation has 512Meg of RAM, so .....
# If you want to make this file smaller, these functions can
# be converted into scripts.
#
# Many functions were taken (almost) straight from the bash-2.04
# examples.
#
#=====

#-----
# Personal Aliases
#-----

alias rm='rm -i'
alias cp='cp -i'
alias mv='mv -i'
# -> Prevents accidentally clobbering files.

alias h='history'
alias j='jobs -l'
alias r='rlogin'
alias which='type -all'
alias ..='cd ..'
alias path='echo -e ${PATH//:/\\n}'
alias print='/usr/bin/lp -o nobanner -d $LPDEST' # Assumes LPDEST is
defined
alias pjet='enscript -h -G -fCourier9 -d $LPDEST' # Pretty-print using
enscript
alias background='xv -root -quit -max -rmode 5' # put a picture in the
background
alias vi='vim'
alias du='du -h'
alias df='df -kh'

# The 'ls' family (this assumes you use the GNU ls)
alias ls='ls -hF --color' # add colors for filetype recognition
alias lx='ls -lXB' # sort by extension
alias lk='ls -lSr' # sort by size
alias la='ls -Al' # show hidden files
alias lr='ls -lR' # recursice ls

```

```

alias lt='ls -ltr' # sort by date
alias lm='ls -al |more' # pipe through 'more'
alias tree='tree -Cs' # nice alternative to 'ls'

# tailoring 'less'
alias more='less'
export PAGER=less
export LESSCHARSET='latin1'
export LESSOPEN='|/usr/bin/lesspipe.sh %s 2>&- ' # Use this if lesspipe.sh
exists
export LESS='-i -N -w -z-4 -g -e -M -X -F -R -P%t?f%f \
:stdin .?pb%pb\%:?lbLine %lb:?bbByte %bb:-... '

# spelling typos - highly personal :- )
alias xs='cd'
alias vf='cd'
alias moer='more'
alias moew='more'
alias kk='ll'

#-----
# a few fun ones
#-----

function xtitle ()
{
  case $TERM in
  *term | rxvt)
    echo -n -e "\033]0;${*\007" ;;
  *) ;;
  esac
}

# aliases...
alias top='xtitle Processes on $HOST && top'
alias make='xtitle Making $(basename $PWD) ; make'
alias ncftp="xtitle ncFTP ; ncftp"

# .. and functions
function man ()
{
  xtitle The $(basename $1|tr -d .[:digit:]) manual
  man -a "$*"
}

function ll(){ ls -l "$@" | egrep "^d" ; ls -lXB "$@" 2>&- | egrep -v
"^d|total " ; }
function xemacs() { { command xemacs -private $* 2>&- & } && disown ; }
function te() # wrapper around xemacs/gnuserv
{
  if [ "$(gnuclient -batch -eval t 2>&-)" == "t" ]; then
    gnuclient -q "$@";
  else
    ( xemacs "$@" & );
  fi
}

#-----
# File & strings related functions:
#-----

```

```

function ff() { find . -name '*'$1'*' ; } # find a file
function fe() { find . -name '*'$1'*' -exec $2 {} \; ; } # find a file and
run $2 on it
function fstr() # find a string in a set of files
{
  if [ "$#" -gt 2 ]; then
    echo "Usage: fstr \"pattern\" [files] "
    return;
  fi
  SMSO=$(tput smso)
  RMSO=$(tput rmso)
  find . -type f -name "${2:-*}" -print | xargs grep -sin "$1" | \
sed "s/$1/$SMSO$1$RMSO/gI"
}

function cuttail() # cut last n lines in file, 10 by default
{
  nlines=${2:-10}
  sed -n -e :a -e "1,${nlines}!{P;N;D;};N;ba" $1
}

function lowercase() # move filenames to lowercase
{
  for file ; do
    filename=${file##*/}
    case "$filename" in
    */*) dirname==${file%/*} ;;
    *) dirname=. ;;
    esac
    nf=$(echo $filename | tr A-Z a-z)
    newname="${dirname}/${nf}"
    if [ "$nf" != "$filename" ]; then
      mv "$file" "$newname"
      echo "lowercase: $file --> $newname"
    else
      echo "lowercase: $file not changed."
    fi
  done
}

function swap() # swap 2 filenames around
{
  local TMPFILE=tmp.$$
  mv $1 $TMPFILE
  mv $2 $1
  mv $TMPFILE $2
}

#-----
# Process/system related functions:
#-----

function my_ps() { ps $@ -u $USER -o pid,%cpu,%mem,bsdtime,command ; }
function pp() { my_ps f | awk '!/awk/ && $0~var' var=${1:-".*"} ; }

# This function is roughly the same as 'killall' on linux
# but has no equivalent (that I know of) on Solaris
function killps() # kill by process name
{
  local pid pname sig="-TERM" # default signal

```

```

if [ "$#" -lt 1 ] || [ "$#" -gt 2 ]; then
    echo "Usage: killps [-SIGNAL] pattern"
    return;
fi
if [ $# = 2 ]; then sig=$1 ; fi
for pid in $(my_ps | awk '!/awk/ && $0~pat { print $1 }' pat=${!#} ) ; do
    pname=$(my_ps | awk '$1~var { print $5 }' var=$pid )
    if ask "Kill process $pid <$pname> with signal $sig?"
    then kill $sig $pid
    fi
done
}

function my_ip() # get IP addresses
{
    MY_IP=$(/sbin/ifconfig ppp0 | awk '/inet/ { print $2 }' | sed -e
s/addr://)
    MY_ISP=$(/sbin/ifconfig ppp0 | awk '/P-t-P/ { print $3 }' | sed -e s/P-t-
P://)
}

function ii() # get current host related info
{
    echo -e "\nYou are logged on ${RED}$HOST"
    echo -e "\nAdditionnal information:$NC " ; uname -a
    echo -e "\n${RED}Users logged on:$NC " ; w -h
    echo -e "\n${RED}Current date :$NC " ; date
    echo -e "\n${RED}Machine stats :$NC " ; uptime
    echo -e "\n${RED}Memory stats :$NC " ; free
    my_ip 2>&- ;
    echo -e "\n${RED}Local IP Address :$NC" ; echo ${MY_IP:-"Not connected"}
    echo -e "\n${RED}ISP Address :$NC" ; echo ${MY_ISP:-"Not connected"}
    echo
}

# Misc utilities:

function repeat() # repeat n times command
{
    local i max
    max=$1; shift;
    for ((i=1; i <= max ; i++)); do # --> C-like syntax
        eval "$@";
    done
}

function ask()
{
    echo -n "$@" '[y/n] ' ; read ans
    case "$ans" in
    y*|Y*) return 0 ;;
    *) return 1 ;;
    esac
}

#=====
#
# PROGRAMMABLE COMPLETION - ONLY SINCE BASH-2.04
# (Most are taken from the bash 2.05 documentation)

```

```

# You will in fact need bash-2.05 for some features
#
#=====

if [ "${BASH_VERSION%.*}" \< "2.05" ]; then
  echo "You will need to upgrade to version 2.05 for programmable
completion"
  return
fi

shopt -s extglob # necessary
set +o nounset # otherwise some completions will fail

complete -A hostname rsh rcp telnet rlogin r ftp ping disk
complete -A command nohup exec eval trace gdb
complete -A command command type which
complete -A export printenv
complete -A variable export local readonly unset
complete -A enabled builtin
complete -A alias alias unalias
complete -A function function
complete -A user su mail finger

complete -A helptopic help # currently same as builtins
complete -A shopt shopt
complete -A stopped -P '%' bg
complete -A job -P '%' fg jobs disown

complete -A directory mkdir rmdir
complete -A directory -o default cd

complete -f -d -X '*.gz' gzip
complete -f -d -X '*.bz2' bzip2
complete -f -o default -X '!*.gz' gunzip
complete -f -o default -X '!*.bz2' bunzip2
complete -f -o default -X '!*.pl' perl perl5
complete -f -o default -X '!*.ps' gs ghostview ps2pdf ps2ascii
complete -f -o default -X '!*.dvi' dvips dvi2pdf xdvi dviselect dvitype
complete -f -o default -X '!*.pdf' acroread pdf2ps
complete -f -o default -X '!*.*(pdf|ps)' gv
complete -f -o default -X '!*.texi*' makeinfo texi2dvi texi2html texi2pdf
complete -f -o default -X '!*.tex' tex latex sltex
complete -f -o default -X '!*.lyx' lyx
complete -f -o default -X '!*.*(jpg|gif|xpm|png|bmp)' xv gimp
complete -f -o default -X '!*.mp3' mpg123
complete -f -o default -X '!*.ogg' ogg123

# This is a 'universal' completion function - it works when commands have
# a so-called 'long options' mode , ie: 'ls --all' instead of 'ls -a'
_universal_func ()
{
  case "$2" in
    -*) ;;
    *) return ;;
  esac

  case "$1" in
    \~*) eval cmd=$1 ;;
    *) cmd="$1" ;;
  esac
}

```

```

COMP_REPLY=( "$cmd" --help | sed -e '/--/!d' -e 's/.*--\([^ ]*\).*/--
\1/' | \
grep ^"$2" |sort -u) )
}
complete -o default -F _universal_func ldd wget bash id info

_make_targets ()
{
    local mdef makef gcmd cur prev i

    COMP_REPLY=()
    cur=${COMP_WORDS[COMP_CWORD]}
    prev=${COMP_WORDS[COMP_CWORD-1]}

    # if prev argument is -f, return possible filename completions.
    # we could be a little smarter here and return matches against
    # `makefile Makefile *.mk', whatever exists
    case "$prev" in
    -*f) COMP_REPLY=( $(compgen -f $cur ) ); return 0;;
    esac

    # if we want an option, return the possible posix options
    case "$cur" in
    -) COMP_REPLY=(-e -f -i -k -n -p -q -r -S -s -t); return 0;;
    esac

    # make reads `makefile' before `Makefile'
    if [ -f makefile ]; then
        mdef=makefile
    elif [ -f Makefile ]; then
        mdef=Makefile
    else
        mdef=*.mk # local convention
    fi

    # before we scan for targets, see if a makefile name was specified
    # with -f
    for (( i=0; i < ${#COMP_WORDS[@]}; i++ )); do
        if [[ ${COMP_WORDS[i]} == -*f ]]; then
            eval makef=${COMP_WORDS[i+1]} # eval for tilde expansion
            break
        fi
    done

    [ -z "$makef" ] && makef=$mdef

    # if we have a partial word to complete, restrict completions to
    # matches of that word
    if [ -n "$2" ]; then gcmd='grep "^$2"' ; else gcmd=cat ; fi

    # if we don't want to use *.mk, we can take out the cat and use
    # test -f $makef and input redirection
    COMP_REPLY=( $(cat $makef 2>/dev/null | awk 'BEGIN {FS=":"} /^[^.# ][^=]*:/
{print $1}' | tr -s ' ' '\012' | sort -u | eval $gcmd ) )
}

complete -F _make_targets -X '+( $*|*.[cho])' make gmake pmake

_configure_func ()
{

```

```

case "$2" in
-*) ;;
*) return ;;
esac

case "$1" in
\~*) eval cmd=$1 ;;
*) cmd="$1" ;;
esac

COMPREPLY=( $($cmd" --help | awk '{if ($1 ~ /--.*/) print $1}' | grep
^"$2" | sort -u) )
}

complete -F _configure_func configure

# cvs(1) completion
_cvs ()
{
local cur prev
COMPREPLY=()
cur=${COMP_WORDS[COMP_CWORD]}
prev=${COMP_WORDS[COMP_CWORD-1]}

if [ $COMP_CWORD -eq 1 ] || [ "${prev:0:1}" = "-" ]; then
COMPREPLY=( $( compgen -W 'add admin checkout commit diff \
export history import log rdiff release remove rtag status \
tag update' $cur ))
else
COMPREPLY=( $( compgen -f $cur ))
fi
return 0
}
complete -F _cvs cvs

_killall ()
{
local cur prev
COMPREPLY=()
cur=${COMP_WORDS[COMP_CWORD]}

# get a list of processes (the first sed evaluation
# takes care of swapped out processes, the second
# takes care of getting the basename of the process)
COMPREPLY=( $( /usr/bin/ps -u $USER -o comm | \
sed -e '1,1d' -e 's#[[]\[]##g' -e 's#^.*/##' | \
awk '{if ($0 ~ /^'$cur'/) print $0}' ))

return 0
}

complete -F _killall killall killps

# Local Variables:
# mode:shell-script
# sh-shell:bash
# End:

```