Configs & Dotfiles

Vishakh Kumar

August 31, 2018

Contents

`https://travis-ci.org/grokkingStuff/configuration.svg?branch=master`

```
#!/usr/bin/env bash
## Description: Connects to remote server and relays local changes made in git repo and
```

```
                ################################################
                #                                              #
                #              Author Information              #
                #                                              #
                # Author: Vishakh Pradeep Kumar                #
                # Email: grokkingStuff@gmail.com on 04-2018    #
                # Current maintainer: Vishakh Pradeep Kumar    #
                ################################################
```

```
##############################################################################################
#                                                                                            #
#                                License Information                                         #
#                                                                                            #
# License: GPLv2, see http://www.fsf.org/licensing/licenses/info/GPLv2.html                  #
# and accompanying license "LICENSE.txt". Redistribution + modification under this           #
```

```
# license permitted.                                                                    #
# If you enclose this script or parts of it in your software, it has to                 #
# be accompanied by the same license (see link) and the place where to get              #
# the recent version of this program: https://testssl.sh                                #
# Don't violate the license.                                                            #
#                                                                                       #
# USAGE WITHOUT ANY WARRANTY, THE SOFTWARE IS PROVIDED "AS IS". USE IT AT               #
# your OWN RISK                                                                         #
#########################################################################################



# SYSTEEM LIBRARY

############################################################
# Displays a list of all flags with their descriptions
# Globals:
#   None
# Arguments:
#   None
# Returns:
#   None
############################################################
function system::usage() {
    echo "$0 usage:" &&               \
      grep "[[:space:]].)\\ ##" "$0" |  \         # Find all line in script that have
      sed 's/##//' |                  \         # Replace all '##' with nothing
      sed -r 's/([a-z])\)/-\1/';              # TODO Can't remember
}
#####################################################################
# Detects the operating system that this script is being run on
# Globals:
#   OSTYPE
# Arguments:
#   None
# Returns:
#   MACHINE
#####################################################################
function system::detect_operating_system() {
```

```
local MACHINE
MACHINE=""

case "$OSTYPE" in

###############################################################################
# *nix systems                                                                #
###############################################################################
    solaris*)
        MACHINE="SOLARIS"                                                    # Do
        ;;
    darwin*)
        MACHINE="OSX"
        ;;
    linux*)
        MACHINE="LINUX"
        ;;
    bsd*)
        MACHINE="BSD"
        ;;
#    aix*)
#        MACHINE="AIX"
#        ;;
#    #Was gonna add AIX but I dunno if it has the $OSTYPE variable and I don't real


###############################################################################
# windows systems                                                             #
###############################################################################
    cygwin*)
        MACHINE="WINDOWS"
        ;&                                                                   # Sin
    msys*)
        MACHINE="WINDOWS"

                                                                             # We
        unameOut="$(uname -s)"
        case "${unameOut}" in
            CYGWIN*)
                MACHINE="WINDOWS-CYGWIN"
```

```
                    # This should work for git shell as well.
                    # I'm not sure why you're using git-shell to do anything except ru
                    ;;
                MINGW32_NT*)
                    MACHINE="WINDOWS-32"
                    ;;
                MINGW64_NT*)
                    MACHINE="WINDOWS-64"
                    ;;
                Linux*)
                    MACHINE="WINDOWS-POWERSHELL"
                    # Not sure why Powershell returns Linux when uname-s is passed to
                    echo "This script will not run in Powershell. Please install a bas
                    echo "Terminating program."
                    exit 1

            esac
            ;;

    ###########################################################################
    # This shouldn't happen but I'm super interested if it does!            #
    ###########################################################################
        *)
            MACHINE="unknown: $OSTYPE"
            echo "I don't know what you're running but I'm interested! Send me an emai
            echo "I'm guessing you're running some sort of custom unix machine so as l
            echo "I mean, seriously, what are you running! Is it a really old system a
            echo "If you do have issues, do send me a email but I can't promise I can
            ;;
    esac

    # Time to return the answer
    return "$MACHINE"
}
############################################################
# Allows for user to send time-tagged strings into STDERR
# Globals:
#   None
# Arguments:
#   Array of String(s)
```

```
# Returns:
#   None
############################################################
function system::err() {
  echo "[$(date +'%Y-%m-%dT%H:%M:%S%z')]: $*" >&2
}
################################################################################
# Checks if the list of commands given to it is executable and available on a system
# Globals:
#   None
# Arguments:
#
# Returns:
#   None
################################################################################
function system::check_required_programs() {
  for p in "${@}"; do
    hash "${p}" 2>&- || \
        { system::err "Required program \"${p}\" not installed or in search PATH.";
          exit 1;
        }
  done
}
################################################################################
## Checks if current folder is a VCS and if so, finds the location of the root reposit
## Globals:
##   None
## Arguments:
##   None
## Returns
##   VCS_REPO_ROOT as String
################################################################################
#function system::vcs_repo_root() {
#
#  local VCS_REPO_ROOT;
#  VCS_REPO_ROOT="";
#
#  # Check if repository is a git repo
#  if git rev-parse --is-inside-work-tree 2> /dev/null; then
#    # This is a valid git repository.
```

```
#    VCS_REPO_ROOT="$(git rev-parse --show-toplevel)";
#
#  elif hg --cwd ./ root 2> /dev/null; then
#    # This is a valid mercurial repository.
#    VCS_REPO_ROOT="$(hg root)";
#
#  elif svn ls ./ > /dev/null; then
#    # This is a valid svn repository.
#    VCS_REPO_ROOT="$(svn info --show-item wc-root)";
#  fi
#
#  if [[ -z VCS_REPO_ROOT ]]; then
#    echo $VCS_REPO_ROOT;
#  else
#    system:err "Current directory is not within a vcs repository.";
#  fi
#}
################################################
## Initialise colour variables and text options
## Global:
##   None
## Arguments:
##   None:
## Returns:
##   None
################################################
#function colour_init() {
#    if [[ -z ${no_colour-} ]]; then
#
#        readonly reset_color="$(tput sgr0 2> /dev/null || true)"
#        # Text attributes
#        readonly ta_bold="$(tput bold 2> /dev/null || true)"
#        printf '%b' "$ta_none"
#        readonly ta_uscore="$(tput smul 2> /dev/null || true)"
#        printf '%b' "$ta_none"
#        readonly ta_blink="$(tput blink 2> /dev/null || true)"
#        printf '%b' "$ta_none"
#        readonly ta_reverse="$(tput rev 2> /dev/null || true)"
#        printf '%b' "$ta_none"
#        readonly ta_conceal="$(tput invis 2> /dev/null || true)"
```

```
#         printf '%b' "$ta_none"
#
#         # Foreground codes
#         readonly fg_black="$(tput setaf 0     2> /dev/null || true)"
#         printf '%b' "$ta_none"
#         readonly fg_blue="$(tput setaf 4      2> /dev/null || true)"
#         printf '%b' "$ta_none"
#         readonly fg_cyan="$(tput setaf 6      2> /dev/null || true)"
#         printf '%b' "$ta_none"
#         readonly fg_green="$(tput setaf 2     2> /dev/null || true)"
#         printf '%b' "$ta_none"
#         readonly fg_magenta="$(tput setaf 5   2> /dev/null || true)"
#         printf '%b' "$ta_none"
#         readonly fg_red="$(tput setaf 1       2> /dev/null || true)"
#         printf '%b' "$ta_none"
#         readonly fg_white="$(tput setaf 7     2> /dev/null || true)"
#         printf '%b' "$ta_none"
#         readonly fg_yellow="$(tput setaf 3    2> /dev/null || true)"
#         printf '%b' "$ta_none"
#
#         # Background codes
#         readonly bg_black="$(tput setab 0     2> /dev/null || true)"
#         printf '%b' "$ta_none"
#         readonly bg_blue="$(tput setab 4      2> /dev/null || true)"
#         printf '%b' "$ta_none"
#         readonly bg_cyan="$(tput setab 6      2> /dev/null || true)"
#         printf '%b' "$ta_none"
#         readonly bg_green="$(tput setab 2     2> /dev/null || true)"
#         printf '%b' "$ta_none"
#         readonly bg_magenta="$(tput setab 5   2> /dev/null || true)"
#         printf '%b' "$ta_none"
#         readonly bg_red="$(tput setab 1       2> /dev/null || true)"
#         printf '%b' "$ta_none"
#         readonly bg_white="$(tput setab 7     2> /dev/null || true)"
#         printf '%b' "$ta_none"
#         readonly bg_yellow="$(tput setab 3    2> /dev/null || true)"
#         printf '%b' "$ta_none"
#     else
#         readonly reset_color=''
#         # Text attributes
```

```
#        readonly ta_bold=''
#        readonly ta_uscore=''
#        readonly ta_blink=''
#        readonly ta_reverse=''
#        readonly ta_conceal=''
#
#        # Foreground codes
#        readonly fg_black=''
#        readonly fg_blue=''
#        readonly fg_cyan=''
#        readonly fg_green=''
#        readonly fg_magenta=''
#        readonly fg_red=''
#        readonly fg_white=''
#        readonly fg_yellow=''
#
#        # Background codes
#        readonly bg_black=''
#        readonly bg_blue=''
#        readonly bg_cyan=''
#        readonly bg_green=''
#        readonly bg_magenta=''
#        readonly bg_red=''
#        readonly bg_white=''
#        readonly bg_yellow=''
#    fi
#}
########################################################
# Makes echo POSIX-compliant while retaining options
# Globals:
#   None
# Arguments:
#   None
# Returns:
#   None
########################################################
function system::echo () (
fmt=%s end=\\n IFS=" "

while [ $# -gt 1 ] ; do
```

```
case "$1" in
[!-]*|-*[!ne]*) break ;;
*ne*|*en*) fmt=%b end= ;;
*n*) end= ;;
*e*) fmt=%b ;;
esac
shift
done

printf "%s%s%s" "$fmt" "$end" "$*"
)

function ok() {
    echo -e "[ok] " "$1"
}

function bot() {
    echo -e "\\[._.]/ - " "$1"
}

function running() {
    echo -en "\\u21d2" "$1" ": "
}

function action() {
    echo -en "\\u21d2 $1..."
}

function warn() {
    echo -e "[warning]" "$1"
}

function error() {
    echo -e "[error] " "$1"
}


# MAIN CONTROL FLOW
function main() {
```

```
#

################################################################################################
   bot "Installing Applications!"
   echo "\
   vocal
   readline-devel sqlite3-devel libbz2-devel zlib-devel libopenssl-devel
   python3-virtualenv
   dropbox
   fish
   bash
   zsh
   chromium
   firefox
   tor
   emacs
   git
   vlc
   htop
   bats" > install.txt

   cat install.txt | while read line; do action "Installing $line"; sudo zypper -iq --g

   rm install.txt

   echo "\n\n"
   bot "Installed applications!"

   bot "Creating Organization!"
   if [ -d "~/Dropbox" ]; then
       dropbox start
       dropbox status

       #touch ~/Dropbox/Projects
       #ln ~/Dropbox/Projects ~/Projects
       #touch ~/Dropbox/Agenda
       #touch ~/Dropbox/Documents
       #ln ~/Dropbox/Documents ~/Documents
       #touch ~/Dropbox/Archive
       #ln ~/Dropbox/Archive ~/Archive
```

```
        #touch ~/Dropbox/Website
        #ln ~/Dropbox/Website ~/Website
        #touch ~/Dropbox/Learning
        #ln ~/Dropbox/Learning ~/Learning
        #touch ~/Dropbox/Medical
        #ln ~/Dropbox/Medical ~/Medical
        #touch ~/Dropbox/AssetManagement
        #ln ~/Dropbox/AssetManagement ~/AssetManagement

        #
    fi
    bot "Created organization!"
################################################################################

# Hello there!
echo "
@test "Test if applications are installed" {
    command -v dropbox
    command -v fish
    command -v bash
    command -v zsh
    command -v chromium
    command -v firefox
    command -v tor
    command -v emacs
    command -v git
    command -v vlc
    command -v htop
    command -v bats
}
@test "Check if pyenv has installed successfully" {
    command -v pyenv
}
@test "Test if the Projects folder exists in the Dropbox folder and in the home directo
 [ -d ~/Dropbox/Projects ]
 [ -d ~/Projects ]
}
@test "Test if the Agenda folder exists in the Dropbox folder and in the home directory
 [ -d ~/Dropbox/Agenda ]
}
```

```
@test "Test if the Documents folder exists in the Dropbox folder and in the home direct
 [ -d ~/Dropbox/Documents ]
 [ -d ~/Documents ]
}
@test "Test if the Configuration folder exists in the Dropbox folder and in the home di
 [ -d ~/Dropbox/Configuration ]
 [ -d ~/Configuration ]
}
@test "Test if the Archive folder exists in the Dropbox folder and in the home director
 [ -d ~/Dropbox/Archive ]
 [ -d ~/Archive ]
}
@test "Test if the Website folder exists in the Dropbox folder and in the home director
 [ -d ~/Dropbox/Website ]
 [ -d ~/Website ]
}
@test "Test if the Learning folder exists in the Dropbox folder and in the home directo
 [ -d ~/Dropbox/Learning ]
 [ -d ~/Learning ]
}
@test "Test if the Medical folder exists in the Dropbox folder and in the home director
 [ -d ~/Dropbox/Medical ]
 [ -d ~/Medical ]
}
@test "Test if the AssetManagement folder exists in the Dropbox folder and in the home
 [ -d ~/Dropbox/AssetManagement ]
 [ -d ~/AssetManagement ]
}
" > test_install.bats
bats test_install.bats

   }

main "$@"
```

Figure 1: The overall structure of connect.sh

0.1 Author Information

Because someone needs to take the blame for when this script goes insane. Seriously, someone take this piece of shit code from me and make it better. Free brownies for whoever does that.

```
#################################################
#                                               #
#              Author Information               #
#                                               #
# Author: Vishakh Pradeep Kumar                 #
# Email: grokkingStuff@gmail.com on 04-2018     #
# Current maintainer: Vishakh Pradeep Kumar     #
#################################################
```

0.1.1 Spacemacs Configuration SPACEMACS

```
(setq user-full-name "Vishakh Kumar"
      user-mail-address "vishakhpradeepkumar@gmail.com")
      ;; calendar-latitude 37.4
      ;; calendar-longitude -122.1
      ;; calendar-location-name "Mountain View, CA")
```

0.2 License information

Even if it seems pretentious, it's good to have a license so that other people can use it. Since this code isn't exactly going to be used in a production environment, I'm going to stick a GPL license on it.

```
#################################################################################
#                                                                               #
#                                License Information                            #
#                                                                               #
# License: GPLv2, see http://www.fsf.org/licensing/licenses/info/GPLv2.html     #
# and accompanying license "LICENSE.txt". Redistribution + modification under this  #
# license permitted.                                                            #
# If you enclose this script or parts of it in your software, it has to         #
# be accompanied by the same license (see link) and the place where to get      #
# the recent version of this program: https://testssl.sh                        #
# Don't violate the license.                                                    #
#                                                                               #
```

```
# USAGE WITHOUT ANY WARRANTY, THE SOFTWARE IS PROVIDED "AS IS". USE IT AT           #
# your OWN RISK                                                                     #
#####################################################################################
```

1 Tests

We'll be interweaving tests with code in this org file and seperating them in files.

```
#!./test/libs/bats/bin/bats

@test "Test if applications are installed" {
    command -v dropbox
    command -v fish
    command -v bash
    command -v zsh
    command -v chromium
    command -v firefox
    command -v tor
    command -v emacs
    command -v git
    command -v vlc
    command -v htop
    command -v bats
}
@test "Check if pyenv has installed successfully" {
    command -v pyenv
}
```

To pull all these submodules into test/libs, run the below from the root of your git repo and commit the result:

```
mkdir -p test/libs

git submodule add https://github.com/bats-core/bats-core test/libs/bats-core
```

1.1 Continuous Integration

We'll be using Travis CI for continuous integration.

```
before_install:
- docker pull opensuse/tumbleweed # Use opensuse Tumbleweed as test
- docker run -d -p 127.0.0.1:80:4567 opensuse/tumbleweed /bin/sh -c "cd /root/sinatra;
- docker ps -a
- docker run opensuse/tumbleweed /bin/sh -c "cd /root/sinatra; bundle exec rake test"



sudo: required
language: bash
services:
  - docker

script:
    - ./test/libs/bats/bin/bats test.bats
```

2 Bash Helper Functions

Bash is a pain in the ass to work with if you need to be safe. This library allows you to write bash that's well-organized, somewhat tested, and hopefully cross platform.

2.1 Preamble

For all the stuff that doesn't really matter to the structure of the program but is quite important for everything else. Most of this should be taken care of by the configBot.

2.1.1 Example of an implementation of getopts and constants that's not bad

```
##########################
# Constants Declaration #
##########################

# Home computer information
USER_VCS_REPO="$(system::vcs_repo_root)"
USER_MACHINE="$(system::detect_operating_system)"

# Remote user information
```

```
REMOTE_IPADDRESS='143.215.98.17'
REMOTE_USER='pi'
REMOTE_USER_PASSWORD='raspberry'
REMOTE_LOCATION='/home/pi/Github/2018'

########################
# User input & Flags  #
########################

while getopts ":iufph:*" o; do
    case "${o}" in

        i) ## IP Address flag. Specify ip address. Default is 143.215.98.17
            REMOTE_IPADDRESS="${OPTARG}"
            ;;

        u) ## Remote username flag. Specify username of raspberry pi. Default is 'pi'
            REMOTE_USER="${OPTARG}"
            ;;

        f) ## Location of remote folder flag. Specify location of github repo on raspbe
            REMOTE_LOCATION="${OPTARG}"
            ;;

        p) ## Password flag. Specify a password for user on remote server
            REMOTE_USER_PASSWORD="${OPTARG}"
            ;;

        h) ## Help flag. Displays flag options
            system::usage
            exit 0
            ;;

        :)  # For when a mandatory argument is skipped.
            system::err "Option -$OPTARG requires an argument."
            system::usage
            exit 1
            ;;
        *)
            system::err "Unexpected option ${flag}"
```

```
            system::usage
            exit 1
            ;;
    esac
done

##############################
# Constants turned read-only #
##############################

# Home computer information
readonly USER_VCS_REPO
readonly USER_MACHINE

# Remote user information
readonly REMOTE_IPADDRESS
readonly REMOTE_USER
readonly REMOTE_USER_PASSWORD
readonly REMOTE_LOCATION
```

Figure 2: Implementation of getopts

2.2 System library LIBRARY:BASH

Functions that are used to query or support the system fall under this library.

- I can't run this in CMD.EXE! What do I do?

  CMD.EXE does not have an inbuilt utility to run sh files. You can install a Linux shell for Windows which should be more than adequate for your purposes. Alternatively, you can install Powershell & Cygwin, although the Linux shell is definitely recommended. Just to be clear, CMD.EXE can run scripts! It's just that no sane man would build a good script in a .cmd file out of his own volition.

- This doesn't run on my OS.

  Huh. That's pretty interesting. This script should run on any system that supports bash (although it may have a few eccentricities.) If you're sure it's not your fault, you should totally send me an email about that.

- This particular function seems too useful for a simple script like this. It's not bad.

  I'm glad you think so! It's really there because I fell down a rabbit hole and I overestimated the importance of being ultra-portable. Use it if you can in your own scripts!

```
# SYSTEEM LIBRARY

############################################################
# Displays a list of all flags with their descriptions
# Globals:
#   None
# Arguments:
#   None
# Returns:
#   None
############################################################
function system::usage() {
    echo "$0 usage:" &&               \
      grep "[[:space:]].)\\ ##" "$0" |  \         # Find all line in script that have
      sed 's/##//' |                  \         # Replace all '##' with nothing
      sed -r 's/([a-z])\)/-\1/';              # TODO Can't remember
}
#####################################################################
# Detects the operating system that this script is being run on
# Globals:
#   OSTYPE
# Arguments:
#   None
# Returns:
#   MACHINE
#####################################################################
function system::detect_operating_system() {

    local MACHINE
    MACHINE=""

    case "$OSTYPE" in
```

```
##############################################################################
# *nix systems                                                               #
##############################################################################
    solaris*)
        MACHINE="SOLARIS"                                                    # Do
        ;;
    darwin*)
        MACHINE="OSX"
        ;;
    linux*)
        MACHINE="LINUX"
        ;;
    bsd*)
        MACHINE="BSD"
        ;;
#    aix*)
#        MACHINE="AIX"
#        ;;
#    #Was gonna add AIX but I dunno if it has the $OSTYPE variable and I don't real


##############################################################################
# windows systems                                                            #
##############################################################################
    cygwin*)
        MACHINE="WINDOWS"
        ;&                                                                   # Si
    msys*)
        MACHINE="WINDOWS"

                                                                             # We
        unameOut="$(uname -s)"
        case "${unameOut}" in
            CYGWIN*)
                MACHINE="WINDOWS-CYGWIN"
                # This should work for git shell as well.
                # I'm not sure why you're using git-shell to do anything except ru
                ;;
            MINGW32_NT*)
                MACHINE="WINDOWS-32"
```

```
                ;;
            MINGW64_NT*)
                MACHINE="WINDOWS-64"
                ;;
            Linux*)
                MACHINE="WINDOWS-POWERSHELL"
                # Not sure why Powershell returns Linux when uname-s is passed to
                echo "This script will not run in Powershell. Please install a bas
                echo "Terminating program."
                exit 1

        esac
        ;;

    ###########################################################################
    # This shouldn't happen but I'm super interested if it does!              #
    ###########################################################################
        *)
            MACHINE="unknown: $OSTYPE"
            echo "I don't know what you're running but I'm interested! Send me an emai
            echo "I'm guessing you're running some sort of custom unix machine so as l
            echo "I mean, seriously, what are you running! Is it a really old system a
            echo "If you do have issues, do send me a email but I can't promise I can
            ;;
    esac

    # Time to return the answer
    return "$MACHINE"
}
#############################################################
# Allows for user to send time-tagged strings into STDERR
# Globals:
#   None
# Arguments:
#   Array of String(s)
# Returns:
#   None
#############################################################
function system::err() {
  echo "[$(date +'%Y-%m-%dT%H:%M:%S%z')]: $*" >&2
```

```
}
################################################################################
# Checks if the list of commands given to it is executable and available on a system
# Globals:
#   None
# Arguments:
#
# Returns:
#   None
################################################################################
function system::check_required_programs() {
  for p in "${@}"; do
    hash "${p}" 2>&- || \
        { system::err "Required program \"${p}\" not installed or in search PATH.";
          exit 1;
        }
  done
}
################################################################################
## Checks if current folder is a VCS and if so, finds the location of the root reposit
## Globals:
##   None
## Arguments:
##   None
## Returns
##   VCS_REPO_ROOT as String
################################################################################
#function system::vcs_repo_root() {
#
#  local VCS_REPO_ROOT;
#  VCS_REPO_ROOT="";
#
#  # Check if repository is a git repo
#  if git rev-parse --is-inside-work-tree 2> /dev/null; then
#    # This is a valid git repository.
#    VCS_REPO_ROOT="$(git rev-parse --show-toplevel)";
#
#  elif hg --cwd ./ root 2> /dev/null; then
#    # This is a valid mercurial repository.
#    VCS_REPO_ROOT="$(hg root)";
```

```
#
#  elif svn ls ./ > /dev/null; then
#    # This is a valid svn repository.
#    VCS_REPO_ROOT="$(svn info --show-item wc-root)";
#  fi
#
#  if [[ -z VCS_REPO_ROOT ]]; then
#    echo $VCS_REPO_ROOT;
#  else
#    system:err "Current directory is not within a vcs repository.";
#  fi
#}
################################################
## Initialise colour variables and text options
## Global:
##   None
## Arguments:
##   None:
## Returns:
##   None
################################################
#function colour_init() {
#    if [[ -z ${no_colour-} ]]; then
#
#        readonly reset_color="$(tput sgr0 2> /dev/null || true)"
#        # Text attributes
#        readonly ta_bold="$(tput bold 2> /dev/null || true)"
#        printf '%b' "$ta_none"
#        readonly ta_uscore="$(tput smul 2> /dev/null || true)"
#        printf '%b' "$ta_none"
#        readonly ta_blink="$(tput blink 2> /dev/null || true)"
#        printf '%b' "$ta_none"
#        readonly ta_reverse="$(tput rev 2> /dev/null || true)"
#        printf '%b' "$ta_none"
#        readonly ta_conceal="$(tput invis 2> /dev/null || true)"
#        printf '%b' "$ta_none"
#
#        # Foreground codes
#        readonly fg_black="$(tput setaf 0     2> /dev/null || true)"
#        printf '%b' "$ta_none"
```

```
#        readonly fg_blue="$(tput setaf 4      2> /dev/null || true)"
#        printf '%b' "$ta_none"
#        readonly fg_cyan="$(tput setaf 6      2> /dev/null || true)"
#        printf '%b' "$ta_none"
#        readonly fg_green="$(tput setaf 2     2> /dev/null || true)"
#        printf '%b' "$ta_none"
#        readonly fg_magenta="$(tput setaf 5   2> /dev/null || true)"
#        printf '%b' "$ta_none"
#        readonly fg_red="$(tput setaf 1       2> /dev/null || true)"
#        printf '%b' "$ta_none"
#        readonly fg_white="$(tput setaf 7     2> /dev/null || true)"
#        printf '%b' "$ta_none"
#        readonly fg_yellow="$(tput setaf 3    2> /dev/null || true)"
#        printf '%b' "$ta_none"
#
#        # Background codes
#        readonly bg_black="$(tput setab 0     2> /dev/null || true)"
#        printf '%b' "$ta_none"
#        readonly bg_blue="$(tput setab 4      2> /dev/null || true)"
#        printf '%b' "$ta_none"
#        readonly bg_cyan="$(tput setab 6      2> /dev/null || true)"
#        printf '%b' "$ta_none"
#        readonly bg_green="$(tput setab 2     2> /dev/null || true)"
#        printf '%b' "$ta_none"
#        readonly bg_magenta="$(tput setab 5   2> /dev/null || true)"
#        printf '%b' "$ta_none"
#        readonly bg_red="$(tput setab 1       2> /dev/null || true)"
#        printf '%b' "$ta_none"
#        readonly bg_white="$(tput setab 7     2> /dev/null || true)"
#        printf '%b' "$ta_none"
#        readonly bg_yellow="$(tput setab 3    2> /dev/null || true)"
#        printf '%b' "$ta_none"
#    else
#        readonly reset_color=''
#        # Text attributes
#        readonly ta_bold=''
#        readonly ta_uscore=''
#        readonly ta_blink=''
#        readonly ta_reverse=''
#        readonly ta_conceal=''
```

```
#
#        # Foreground codes
#        readonly fg_black=''
#        readonly fg_blue=''
#        readonly fg_cyan=''
#        readonly fg_green=''
#        readonly fg_magenta=''
#        readonly fg_red=''
#        readonly fg_white=''
#        readonly fg_yellow=''
#
#        # Background codes
#        readonly bg_black=''
#        readonly bg_blue=''
#        readonly bg_cyan=''
#        readonly bg_green=''
#        readonly bg_magenta=''
#        readonly bg_red=''
#        readonly bg_white=''
#        readonly bg_yellow=''
#    fi
#}
########################################################
# Makes echo POSIX-compliant while retaining options
# Globals:
#   None
# Arguments:
#   None
# Returns:
#   None
########################################################
function system::echo () (
fmt=%s end=\\n IFS=" "

while [ $# -gt 1 ] ; do
case "$1" in
[!-]*|-*[!ne]*) break ;;
*ne*|*en*) fmt=%b end= ;;
*n*) end= ;;
*e*) fmt=%b ;;
```

```
esac
shift
done

printf "%s%s%s" "$fmt" "$end" "$*"
)

function ok() {
    echo -e "[ok] " "$1"
}

function bot() {
    echo -e "\\[._.]/ - " "$1"
}

function running() {
    echo -en "\\u21d2" "$1" ": "
}

function action() {
    echo -en "\\u21d2 $1..."
}

function warn() {
    echo -e "[warning]" "$1"
}

function error() {
    echo -e "[error] " "$1"
}
```

2.2.1 Help prompt

A quick and effective help function that uses the comments in the flag case block. Scans this file for a "##" in front of a ")" and displays those lines exclusively. Restrict comments to single # to avoid unnecessary mixup.

```
#########################################################
# Displays a list of all flags with their descriptions
# Globals:
```

```
#   None
# Arguments:
#   None
# Returns:
#   None
########################################################
function system::usage() {
    echo "$0 usage:" &&               \
      grep "[[:space:]].)\\ ##" "$0" |  \         # Find all line in script that have
      sed 's/##//' |                  \          # Replace all '##' with nothing
      sed -r 's/([a-z])\)/-\1/';              # TODO Can't remember
}
```

2.2.2 Detect operating system FUNCTION:BASH

Since this command will be executed by different people of multiple operating systems, I've decided to use as many bash built-ins as possible for portability. However, there are still things that need to be set for each operating system. This code block detects the operating system and makes it available in the variable $MACHINE. I was gonna hack together a way to do this using the uname command but I think using pre-defined $OSTYPE variable is cleaner.

```
#####################################################################
# Detects the operating system that this script is being run on
# Globals:
#   OSTYPE
# Arguments:
#   None
# Returns:
#   MACHINE
#####################################################################
function system::detect_operating_system() {

    local MACHINE
    MACHINE=""

    case "$OSTYPE" in

    ###########################################################################
    # *nix systems                                                            #
```

```
##############################################################################
    solaris*)
        MACHINE="SOLARIS"                                                    # Do
        ;;
    darwin*)
        MACHINE="OSX"
        ;;
    linux*)
        MACHINE="LINUX"
        ;;
    bsd*)
        MACHINE="BSD"
        ;;
#    aix*)
#        MACHINE="AIX"
#        ;;
#    #Was gonna add AIX but I dunno if it has the $OSTYPE variable and I don't real


##############################################################################
# windows systems                                                            #
##############################################################################
    cygwin*)
        MACHINE="WINDOWS"
        ;&                                                                   # Sim
    msys*)
        MACHINE="WINDOWS"

                                                                             # We
        unameOut="$(uname -s)"
        case "${unameOut}" in
            CYGWIN*)
                MACHINE="WINDOWS-CYGWIN"
                # This should work for git shell as well.
                # I'm not sure why you're using git-shell to do anything except run
                ;;
            MINGW32_NT*)
                MACHINE="WINDOWS-32"
                ;;
            MINGW64_NT*)
```

```
                    MACHINE="WINDOWS-64"
                    ;;
                Linux*)
                    MACHINE="WINDOWS-POWERSHELL"
                    # Not sure why Powershell returns Linux when uname-s is passed to
                    echo "This script will not run in Powershell. Please install a bas
                    echo "Terminating program."
                    exit 1

            esac
            ;;

    ###########################################################################
    # This shouldn't happen but I'm super interested if it does!            #
    ###########################################################################
        *)
            MACHINE="unknown: $OSTYPE"
            echo "I don't know what you're running but I'm interested! Send me an emai
            echo "I'm guessing you're running some sort of custom unix machine so as l
            echo "I mean, seriously, what are you running! Is it a really old system a
            echo "If you do have issues, do send me a email but I can't promise I can
            ;;
    esac

    # Time to return the answer
    return "$MACHINE"
}
```

Figure 3: bash function to detect the operating system the shell is running on.

2.2.3 Sending time-tagged strings into STDERRfunction:bash

All error messages should go to STDERR (standard error), including user defined errors. This function attaches a date and time to a string and passes it to STDERR Reference: Google Style Sheet: STDOUT vs STDERR

```
############################################################
# Allows for user to send time-tagged strings into STDERR
# Globals:
```

```
#   None
# Arguments:
#   Array of String(s)
# Returns:
#   None
############################################################
function system::err() {
  echo "[$(date +'%Y-%m-%dT%H:%M:%S%z')]: $*" >&2
}
```

Figure 4: Function to generate errors and logs with attached date and time.

2.2.4 Check if required programs are installed FUNCTION:BASH

While this should ideally be taken care of by testing on different systems and by using portable bash builtins, there really isn't a substitute to checking if the command/program you're looking for is installed on the computer.

```
###############################################################################################
# Checks if the list of commands given to it is executable and available on a system
# Globals:
#   None
# Arguments:
#
# Returns:
#   None
###############################################################################################
function system::check_required_programs() {
  for p in "${@}"; do
    hash "${p}" 2>&- || \
        { system::err "Required program \"${p}\" not installed or in search PATH.";
          exit 1;
        }
  done
}
```

2.2.5 Detect VCS system and find root directory FUNCTION:BASH

So it turns out that different VCS have different ways of querying for the location of the root folder. Since I've only used git and I've dabbled in

Mercurial, this code might be outdated and downright wrong. However, gonna stick this in here since it might be handy.

```
##############################################################################
# Checks if current folder is a VCS and if so, finds the location of the root repositor
# Globals:
#   None
# Arguments:
#   None
# Returns
#   VCS_REPO_ROOT as String
##############################################################################
function system::vcs_repo_root() {

  local VCS_REPO_ROOT;
  VCS_REPO_ROOT="";

  # Check if repository is a git repo
  if git rev-parse --is-inside-work-tree 2> /dev/null; then
    # This is a valid git repository.
    VCS_REPO_ROOT="$(git rev-parse --show-toplevel)";

  elif hg --cwd ./ root 2> /dev/null; then
    # This is a valid mercurial repository.
    VCS_REPO_ROOT="$(hg root)";

  elif svn ls ./ > /dev/null; then
    # This is a valid svn repository.
    VCS_REPO_ROOT="$(svn info --show-item wc-root)";
  fi

  if [[ -z VCS_REPO_ROOT ]]; then
    echo $VCS_REPO_ROOT;
  else
    system:err "Current directory is not within a vcs repository.";
  fi
}
```

Figure 5: Function to return root of vcs repository when possible

2.2.6 Colors & Text attributes FUNCTION:CONSTANT:BASH

Because all the colors and fancy effects! Shamelessly stolen from `https://github.com/ralish/bash-script-template/blob/stable/template.sh`

Table 1: Colors available for tput

Num	Colour	#define	R G B
0	black	$\text{COLOR}_{\text{BLACK}}$	0,0,0
1	red	$\text{COLOR}_{\text{RED}}$	1,0,0
2	green	$\text{COLOR}_{\text{GREEN}}$	0,1,0
3	yellow	$\text{COLOR}_{\text{YELLOW}}$	1,1,0
4	blue	$\text{COLOR}_{\text{BLUE}}$	0,0,1
5	magenta	$\text{COLOR}_{\text{MAGENTA}}$	1,0,1
6	cyan	$\text{COLOR}_{\text{CYAN}}$	0,1,1
7	white	$\text{COLOR}_{\text{WHITE}}$	1,1,1

```
################################################
# Initialise colour variables and text options
# Global:
#   None
# Arguments:
#   None:
# Returns:
#   None
################################################
function colour_init() {
    if [[ -z ${no_colour-} ]]; then

        readonly reset_color="$(tput sgr0 2> /dev/null || true)"
        # Text attributes
        readonly ta_bold="$(tput bold 2> /dev/null || true)"
        printf '%b' "$ta_none"
        readonly ta_uscore="$(tput smul 2> /dev/null || true)"
        printf '%b' "$ta_none"
        readonly ta_blink="$(tput blink 2> /dev/null || true)"
        printf '%b' "$ta_none"
        readonly ta_reverse="$(tput rev 2> /dev/null || true)"
        printf '%b' "$ta_none"
        readonly ta_conceal="$(tput invis 2> /dev/null || true)"
```

```
    printf '%b' "$ta_none"

    # Foreground codes
    readonly fg_black="$(tput setaf 0     2> /dev/null || true)"
    printf '%b' "$ta_none"
    readonly fg_blue="$(tput setaf 4      2> /dev/null || true)"
    printf '%b' "$ta_none"
    readonly fg_cyan="$(tput setaf 6      2> /dev/null || true)"
    printf '%b' "$ta_none"
    readonly fg_green="$(tput setaf 2     2> /dev/null || true)"
    printf '%b' "$ta_none"
    readonly fg_magenta="$(tput setaf 5   2> /dev/null || true)"
    printf '%b' "$ta_none"
    readonly fg_red="$(tput setaf 1       2> /dev/null || true)"
    printf '%b' "$ta_none"
    readonly fg_white="$(tput setaf 7     2> /dev/null || true)"
    printf '%b' "$ta_none"
    readonly fg_yellow="$(tput setaf 3    2> /dev/null || true)"
    printf '%b' "$ta_none"

    # Background codes
    readonly bg_black="$(tput setab 0     2> /dev/null || true)"
    printf '%b' "$ta_none"
    readonly bg_blue="$(tput setab 4      2> /dev/null || true)"
    printf '%b' "$ta_none"
    readonly bg_cyan="$(tput setab 6      2> /dev/null || true)"
    printf '%b' "$ta_none"
    readonly bg_green="$(tput setab 2     2> /dev/null || true)"
    printf '%b' "$ta_none"
    readonly bg_magenta="$(tput setab 5   2> /dev/null || true)"
    printf '%b' "$ta_none"
    readonly bg_red="$(tput setab 1       2> /dev/null || true)"
    printf '%b' "$ta_none"
    readonly bg_white="$(tput setab 7     2> /dev/null || true)"
    printf '%b' "$ta_none"
    readonly bg_yellow="$(tput setab 3    2> /dev/null || true)"
    printf '%b' "$ta_none"
else
    readonly reset_color=''
    # Text attributes
```

```
        readonly ta_bold=''
        readonly ta_uscore=''
        readonly ta_blink=''
        readonly ta_reverse=''
        readonly ta_conceal=''

        # Foreground codes
        readonly fg_black=''
        readonly fg_blue=''
        readonly fg_cyan=''
        readonly fg_green=''
        readonly fg_magenta=''
        readonly fg_red=''
        readonly fg_white=''
        readonly fg_yellow=''

        # Background codes
        readonly bg_black=''
        readonly bg_blue=''
        readonly bg_cyan=''
        readonly bg_green=''
        readonly bg_magenta=''
        readonly bg_red=''
        readonly bg_white=''
        readonly bg_yellow=''
    fi
}
```

2.2.6.1 colors$_{\text{textattributes}}$ **CONSTANT:BASH** Text attributes can be changed by writing "ta_" followed by the particular text attribute you want. The options are:

```
# Text attributes
readonly ta_bold="$(tput bold 2> /dev/null || true)"
printf '%b' "$ta_none"
readonly ta_uscore="$(tput smul 2> /dev/null || true)"
printf '%b' "$ta_none"
readonly ta_blink="$(tput blink 2> /dev/null || true)"
printf '%b' "$ta_none"
readonly ta_reverse="$(tput rev 2> /dev/null || true)"
```

Table 2: Different text attribute options

Command	Description
tput bold	# Select bold mode
tput dim	# Select dim (half-bright) mode
tput smul	# Enable underline mode
tput rmul	# Disable underline mode
tput rev	# Turn on reverse video mode
tput smso	# Enter standout (bold) mode
tput rmso	# Exit standout mode

```
printf '%b' "$ta_none"
readonly ta_conceal="$(tput invis 2> /dev/null || true)"
printf '%b' "$ta_none"
```

Table 3: Colors available for tput

Num	Colour	#define	R G B
0	black	$\text{COLOR}_{\text{BLACK}}$	0,0,0
1	red	$\text{COLOR}_{\text{RED}}$	1,0,0
2	green	$\text{COLOR}_{\text{GREEN}}$	0,1,0
3	yellow	$\text{COLOR}_{\text{YELLOW}}$	1,1,0
4	blue	$\text{COLOR}_{\text{BLUE}}$	0,0,1
5	magenta	$\text{COLOR}_{\text{MAGENTA}}$	1,0,1
6	cyan	$\text{COLOR}_{\text{CYAN}}$	0,1,1
7	white	$\text{COLOR}_{\text{WHITE}}$	1,1,1

2.2.6.2 $\text{colors}_{\text{foreground}}$ CONSTANT:BASH

```
# Foreground codes
readonly fg_black="$(tput setaf 0     2> /dev/null || true)"
printf '%b' "$ta_none"
readonly fg_blue="$(tput setaf 4      2> /dev/null || true)"
printf '%b' "$ta_none"
readonly fg_cyan="$(tput setaf 6      2> /dev/null || true)"
printf '%b' "$ta_none"
readonly fg_green="$(tput setaf 2     2> /dev/null || true)"
printf '%b' "$ta_none"
readonly fg_magenta="$(tput setaf 5   2> /dev/null || true)"
printf '%b' "$ta_none"
```

```
readonly fg_red="$(tput setaf 1       2> /dev/null || true)"
printf '%b' "$ta_none"
readonly fg_white="$(tput setaf 7     2> /dev/null || true)"
printf '%b' "$ta_none"
readonly fg_yellow="$(tput setaf 3    2> /dev/null || true)"
printf '%b' "$ta_none"
```

Table 4: Colors available for tput

Num	Colour	#define	R G B
0	black	$\text{COLOR}_{\text{BLACK}}$	0,0,0
1	red	$\text{COLOR}_{\text{RED}}$	1,0,0
2	green	$\text{COLOR}_{\text{GREEN}}$	0,1,0
3	yellow	$\text{COLOR}_{\text{YELLOW}}$	1,1,0
4	blue	$\text{COLOR}_{\text{BLUE}}$	0,0,1
5	magenta	$\text{COLOR}_{\text{MAGENTA}}$	1,0,1
6	cyan	$\text{COLOR}_{\text{CYAN}}$	0,1,1
7	white	$\text{COLOR}_{\text{WHITE}}$	1,1,1

2.2.6.3 $\text{colors}_{\text{background}}$ CONSTANT:BASH

```
# Background codes
readonly bg_black="$(tput setab 0     2> /dev/null || true)"
printf '%b' "$ta_none"
readonly bg_blue="$(tput setab 4      2> /dev/null || true)"
printf '%b' "$ta_none"
readonly bg_cyan="$(tput setab 6      2> /dev/null || true)"
printf '%b' "$ta_none"
readonly bg_green="$(tput setab 2     2> /dev/null || true)"
printf '%b' "$ta_none"
readonly bg_magenta="$(tput setab 5   2> /dev/null || true)"
printf '%b' "$ta_none"
readonly bg_red="$(tput setab 1       2> /dev/null || true)"
printf '%b' "$ta_none"
readonly bg_white="$(tput setab 7     2> /dev/null || true)"
printf '%b' "$ta_none"
readonly bg_yellow="$(tput setab 3    2> /dev/null || true)"
printf '%b' "$ta_none"
```

2.2.6.4 colors$_{\text{nullvalues}}$ CONSTANT:BASH

If we don't use colors in our code but still put references to it in our code, it might cause annoying issues. We'll be setting them to '' so that nothing happens and our code is safe.

```
# Text attributes
readonly ta_bold=''
readonly ta_uscore=''
readonly ta_blink=''
readonly ta_reverse=''
readonly ta_conceal=''

# Foreground codes
readonly fg_black=''
readonly fg_blue=''
readonly fg_cyan=''
readonly fg_green=''
readonly fg_magenta=''
readonly fg_red=''
readonly fg_white=''
readonly fg_yellow=''

# Background codes
readonly bg_black=''
readonly bg_blue=''
readonly bg_cyan=''
readonly bg_green=''
readonly bg_magenta=''
readonly bg_red=''
readonly bg_white=''
readonly bg_yellow=''
```

2.2.7 POSIX compliant echo FUNCTION:BASH

While echo is a rather common tool, it's actually terribly designed. It's only portable if you don't any use flags and it's output isn't consistent. We'll be using printf instead, which is POSIX-compliant and much better designed. As a special function, it will be listed as both system::echo and echo, for ease of use.

```
#######################################################
```

```
# Makes echo POSIX-compliant while retaining options
# Globals:
#   None
# Arguments:
#   None
# Returns:
#   None
########################################################
function system::echo () (
fmt=%s end=\\n IFS=" "

while [ $# -gt 1 ] ; do
case "$1" in
[!-]*|-*[!ne]*) break ;;
*ne*|*en*) fmt=%b end= ;;
*n*) end= ;;
*e*) fmt=%b ;;
esac
shift
done

printf "%s%s%s" "$fmt" "$end" "$*"
)

function ok() {
    echo -e "[ok] " "$1"
}

function bot() {
    echo -e "\\[._.]/ - " "$1"
}

function running() {
    echo -en "\\u21d2" "$1" ": "
}

function action() {
    echo -en "\\u21d2 $1..."
}
```

```
function warn() {
    echo -e "[warning]" "$1"
}

function error() {
    echo -e "[error] " "$1"
}
```

3 Organization

```
if [ -d "~/Dropbox" ]; then
    dropbox start
    dropbox status

    #touch ~/Dropbox/Projects
    #ln ~/Dropbox/Projects ~/Projects
    #touch ~/Dropbox/Agenda
    #touch ~/Dropbox/Documents
    #ln ~/Dropbox/Documents ~/Documents
    #touch ~/Dropbox/Archive
    #ln ~/Dropbox/Archive ~/Archive
    #touch ~/Dropbox/Website
    #ln ~/Dropbox/Website ~/Website
    #touch ~/Dropbox/Learning
    #ln ~/Dropbox/Learning ~/Learning
    #touch ~/Dropbox/Medical
    #ln ~/Dropbox/Medical ~/Medical
    #touch ~/Dropbox/AssetManagement
    #ln ~/Dropbox/AssetManagement ~/AssetManagement

    #
fi
```

3.1 Dropbox

3.1.1 Installation INSTALL

```
dropbox
```

3.2 Folder Organization

3.2.1 Projects

```
touch ~/Dropbox/Projects
ln ~/Dropbox/Projects ~/Projects

@test "Test if the Projects folder exists in the Dropbox folder and in the home directo
 [ -d ~/Dropbox/Projects ]
 [ -d ~/Projects ]
}
```

3.2.2 Agenda

```
touch ~/Dropbox/Agenda

@test "Test if the Agenda folder exists in the Dropbox folder and in the home directory
 [ -d ~/Dropbox/Agenda ]
}
```

3.2.3 Documents

```
touch ~/Dropbox/Documents
ln ~/Dropbox/Documents ~/Documents

@test "Test if the Documents folder exists in the Dropbox folder and in the home direct
 [ -d ~/Dropbox/Documents ]
 [ -d ~/Documents ]
}
```

3.2.4 Configuration

- org-agenda integration

```
(setq org-agenda-files
    (file-expand-wildcards "~/Proposals/*.org")
    (file-expand-wildcards "~/Projects/*.org")
    (file-expand-wildcards "~/PersonalDevelopment/*.org")
    (file-expand-wildcards "~/College/*.org")
    (file-expand-wildcards "~/Business/*.org")
    (file-expand-wildcards "~/Finances/*.org")
)
```

```
#+END_SRC emacs-lisp
```

```
#+NAME: organization_folder
#+BEGIN_SRC sh
touch ~/Dropbox/Configuration
ln ~/Dropbox/Configuration ~/Configuration

@test "Test if the Configuration folder exists in the Dropbox folder and in the home di
 [ -d ~/Dropbox/Configuration ]
 [ -d ~/Configuration ]
}
```

3.2.5 Archive

```
touch ~/Dropbox/Archive
ln ~/Dropbox/Archive ~/Archive

@test "Test if the Archive folder exists in the Dropbox folder and in the home directo
 [ -d ~/Dropbox/Archive ]
 [ -d ~/Archive ]
}
```

3.2.6 Website

```
touch ~/Dropbox/Website
ln ~/Dropbox/Website ~/Website

@test "Test if the Website folder exists in the Dropbox folder and in the home directo
 [ -d ~/Dropbox/Website ]
 [ -d ~/Website ]
}
```

3.2.7 Learning

```
touch ~/Dropbox/Learning
ln ~/Dropbox/Learning ~/Learning

@test "Test if the Learning folder exists in the Dropbox folder and in the home directo
 [ -d ~/Dropbox/Learning ]
 [ -d ~/Learning ]
}
```

3.2.8 Medical

```
touch ~/Dropbox/Medical
ln ~/Dropbox/Medical ~/Medical

@test "Test if the Medical folder exists in the Dropbox folder and in the home directo
 [ -d ~/Dropbox/Medical ]
 [ -d ~/Medical ]
}
```

3.2.9 Asset Management

```
touch ~/Dropbox/AssetManagement
ln ~/Dropbox/AssetManagement ~/AssetManagement

@test "Test if the AssetManagement folder exists in the Dropbox folder and in the home
 [ -d ~/Dropbox/AssetManagement ]
 [ -d ~/AssetManagement ]
}
```

3.2.10 Contacts

4 Applications

In this section, we'll be listing the application name and general info, its package name for our package manager to install it, and any configuration files related to said software.

This allows us to create a list of all applications that we'll need in a single file while keeping them all nice and organized in seperate categories. Keep in mind that programming languages are not included in this section (they have special requirements for a proper development environment) but applications that are installed using a language's package manager belong here.

- **Conventions**
  - Any headline that's an application must have the application tag.
    - If the application name is not immediately indicative of its purpose, a brief description of its type can be included after a hypen.

- Any installation code block in this section should have the tag :install:, headline Installation and name 'install' (install_ if you don't want it to be tested.)
- All configuration files must have a parent headline called 'Configuration' with tag :configuration:
  - If the configuration file is worthy of it's own org file, a link shall be provided for the same.
- If an application is installed with a programming language's package manager, use an appropriate tag and src block name.

	Language	tag	src block name
*	Python 2	python2	$\text{python2}_{\text{install}}$
	Python 3	python3	$\text{python3}_{\text{install}}$

```
** General application category
*** Application name - type of application (if required)        :application:
**** Installation
#+NAME: install               # install_ if you don't want it to be tested
#+BEGIN_SRC sh :padline no :tangle no :noweb yes

#+END_SRC

echo "\
vocal
readline-devel sqlite3-devel libbz2-devel zlib-devel libopenssl-devel
python3-virtualenv
dropbox
fish
bash
zsh
chromium
firefox
tor
emacs
git
vlc
htop
bats" > install.txt

cat install.txt | while read line; do action "Installing $line"; sudo zypper -iq --gpg
```

```
rm install.txt

echo "\n\n"

@test "Test if applications are installed" {
    command -v dropbox
    command -v fish
    command -v bash
    command -v zsh
    command -v chromium
    command -v firefox
    command -v tor
    command -v emacs
    command -v git
    command -v vlc
    command -v htop
    command -v bats
}
```

4.1 Terminal Emulators

Plenty of shells for a hermit crab to choose. I'm going with fish for my interactive shell and bash for my scripts. Will try zsh for specific types of repositories.

4.1.1 fish APPLICATION

4.1.1.1 Installation INSTALL

```
fish
```

4.1.2 bash APPLICATION

4.1.2.1 Installation INSTALL While you shouldn't really have to install bash on a system (since it should just be there), I'm adding this for the sake of completionists everywhere.

```
bash
```

4.1.2.2 Configuration CONFIGURATION Home is where ~~the heart is~~ your aliases are

4.1.2.2.1 Navigation

- Easier navigation: .., ..., ..., and .....

```
alias ..="cd .."
alias ...="cd ../.."
alias ....="cd ../../.."
alias .....="cd ../../../.."
```

- Shortcuts to commonly used folders

```
alias downloads="cd ~/Downloads"
alias desktop="cd ~/Desktop"
alias projects="cd ~/Projects"
```

- Shortcuts to commonly used commands

```
alias g="git"
alias h="history"
```

4.1.2.2.2 grep

- Always enable colored ‘grep‘ output

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

4.1.2.2.3 Enable aliases to be sudo'ed

```
alias sudo='sudo '
```

4.1.2.2.4 Get week number

```
alias week='date +%V'
```

4.1.2.2.5 Stopwatch

```
alias timer='echo "Timer started. Stop with Ctrl-D." && date && time cat && date'
```

4.1.2.2.6 Encryption

- OS X has no ‘md5sum‘, so use ‘md5‘ as a fallback

  ```
  command -v md5sum > /dev/null || alias md5sum="md5"
  ```

- OS X has no ‘sha1sum‘, so use ‘shasum‘ as a fallback

  ```
  command -v sha1sum > /dev/null || alias sha1sum="shasum"
  ```

- Canonical hex dump; some systems have this symlinked

  ```
  command -v hd > /dev/null || alias hd="hexdump -C"
  ```

4.1.2.2.7 Intuitive map function

```
alias map="xargs -n1"
```

4.1.2.2.8 One of @janmoesen's ProTip™s

```
for method in GET HEAD POST PUT DELETE TRACE OPTIONS; do
  alias "$method"="lwp-request -m '$method'"
done
```

4.1.2.2.9 Fun Stuff

- Stuff I never really use but cannot delete either because of `http://xkcd.com/530/`

  ```
  alias stfu="osascript -e 'set volume output muted true'"
  alias pumpitup="osascript -e 'set volume 7'"
  ```

- Starwars Don't remember who showed me this in the fifth grade but it's awesome and it stuck. Thanks!

  ```
  alias starwars="telnet towel.blinkenlights.nl"
  ```

4.1.3 zsh APPLICATION

4.1.3.1 Installation INSTALL

`zsh`

4.2 Browsers

4.2.1 Chromium APPLICATION

4.2.1.1 Installation INSTALL

```
chromium
```

4.2.2 Firefox APPLICATION

4.2.2.1 Installation INSTALL

```
firefox
```

4.2.3 Tor APPLICATION

4.2.3.1 Installation INSTALL

```
tor
```

4.3 Text editors

4.3.1 Emacs APPLICATION

4.3.1.1 Installation INSTALL

```
emacs
```

4.4 cURL configurations options

```
https://curl.haxx.se/docs/manpage.html
```

4.4.1 Limit the time (in seconds) the connection is allowed to take.

```
connect-timeout = 60
```

4.4.2 Follow HTTP redirects.

```
location
```

4.4.3 Display progress as a simple progress bar.

```
progress-bar
```

4.4.4 Show error messages.

```
show-error
```

4.4.5 Send a fake UA string for the HTTP servers that sniff it.

```
user-agent = "Mozilla/5.0 Gecko"
```

4.5 Version Control

4.5.1 Git APPLICATION

4.5.1.1 Installation INSTALL

```
git
```

4.5.1.2 Spacemacs Layer SPACEMACS

```
;;;;;;;;;;;;;;;;;;;;;;;;
;; git version control ;;
;;;;;;;;;;;;;;;;;;;;;;;;
git                     ;;
github                  ;;
magit                   ;;
;;;;;;;;;;;;;;;;;;;;;;;;
```

4.5.1.3 Configuration CONFIGURATION

4.6 Media

4.6.1 VLC - Video Player APPLICATION

4.6.1.1 Installation INSTALL

```
vlc
```

4.6.2 Vocal - Podcast Client APPLICATION

4.6.2.1 Installation INSTALL

```
vocal
```

4.6.3 youtube-dl - Downloader for youtube videos APPLICATION

4.6.3.1 Installation PYTHON2:INSTALL

```
youtube-dl
```

4.7 Activity Monitor

4.7.1 htop APPLICATION

4.7.1.1 Installation INSTALL

```
htop
```

4.7.1.2 Configuration CONFIGURATION All configuration options are located in the .htoprc file. Stolen from god knows where - seems like everyone uses it.

```
# Beware! This file is rewritten every time htop exits.
# The parser is also very primitive, and not human-friendly.
# (I know, it's in the todo list).
fields=0 48 17 18 38 39 40 2 46 47 49 1
sort_key=46
sort_direction=1
hide_threads=0
hide_kernel_threads=1
hide_userland_threads=0
shadow_other_users=0
highlight_base_name=0
highlight_megabytes=1
highlight_threads=0
tree_view=0
header_margin=1
detailed_cpu_time=1
color_scheme=0
delay=15
left_meters=Hostname Tasks LoadAverage Uptime Memory Memory Swap CPU CPU
left_meter_modes=2 2 2 2 1 2 1 1 2
right_meters=AllCPUs
right_meter_modes=1
```

4.8 Communication

4.8.1 Slack

4.8.1.1 Spacemacs Layer SPACEMACS

```
;; There's no escaping the beast

;;;;;;;;;;;;;;;;;;;;;;;
;; Team Communication ;;
;;;;;;;;;;;;;;;;;;;;;;;
slack                 ;;
;;;;;;;;;;;;;;;;;;;;;;;
```

4.8.2 Twitter

4.8.2.1 Spacemacs Layer SPACEMACS

```
;; Because Twitter is addictive

;;;;;;;;;;;;;;;;;;
;; Social Media ;;
;;;;;;;;;;;;;;;;;;
twitter         ;;
;;;;;;;;;;;;;;;;;;
```

4.8.3 Email

4.8.3.1 Spacemacs Layer SPACEMACS

```
;; Decent email client

;;;;;;;;;;;;;;;;;;
;; Email client ;;
;;;;;;;;;;;;;;;;;;
mu4e            ;;
;;;;;;;;;;;;;;;;;;
```

4.8.4 RSS

```
;; RSS - clinginging on to Web 2.0

;;;;;;;;;;;;;;;;
;; RSS client ;;
;;;;;;;;;;;;;;;;
elfeed        ;;
;;;;;;;;;;;;;;;;
```

4.9 Documents

```
;; RSS - clinginging on to Web 2.0

;;;;;;;;;;;;;;;;;;;
;; pdf utilities ;;
;;;;;;;;;;;;;;;;;;;
pdf-tools        ;;
;;;;;;;;;;;;;;;;;;;
```

4.10 File manager

4.10.1 ranger

```
;; RSS - clinginging on to Web 2.0

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; pdf utilities               ;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
(ranger :variables             ;;
        ranger-show-preview t) ;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

5 Python

```
#########
# Pyenv #
#########
```

```
# Taken from https://www.reddit.com/r/openSUSE/comments/70ozge/using_multiple_python_ve

git clone https://github.com/pyenv/pyenv.git ~/.pyenv
echo 'export PYENV_ROOT="$HOME/.pyenv"' >> ~/.bashrc
echo 'export PATH="$PYENV_ROOT/bin:$PATH"' >> ~/.bashrc
echo -e 'if command -v pyenv 1>/dev/null 2>&1; then\n  eval "$(pyenv init -)"\nfi' >> 
pyenv install 3.6.0
pyenv install 2.7.13
# All virtualenvs will be on...
# export WORKON_HOME=~/.ve
mkdir -p ~/.ve

# All projects will be on...
# export PROJECT_HOME=~/Projects
mkdir -p ~/Projects

# The -p flag is in case these folders have been created earlier - without it, mkdir re
pyenv virtualenv 3.6.0 jupyter3
pyenv virtualenv 3.6.0 tools3
pyenv virtualenv 2.7.13 ipython2
pyenv virtualenv 2.7.13 tools2
pyenv activate jupyter3
pip install jupyter
python -m ipykernel install --user
pyenv deactivate
pyenv activate ipython2
pip install ipykernel
python -m ipykernel install --user
pyenv deactivate
pyenv activate tools3
pip install youtube-dl gnucash-to-beancount rows
pyenv deactivate
pyenv activate tools2
pip install rename s3cmd fabric mercurial
pyenv deactivate
pyenv global 3.6.0 2.7.13 jupyter3 ipython2 tools3 tools2
ipython profile create
curl -L http://hbn.link/hb-ipython-startup-script > ~/.ipython/profile_default/startup/
```

5.1 Spacemacs Layer SPACEMACS

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; python layer configuration                                          ;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
(python :variables                                                     ;;
        python-sort-imports-on-save t                                  ;;
        python-test-runner 'pytest                                     ;;
        :packages                                                      ;;
        (not hy-mode)  ; I maintain local 'hy-mode'                    ;;
        (not importmagic))  ; Broken? Don't need it.                   ;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

5.2 Pyenv

pyenv is used to isolate Python versions. For example, you may want to test your code against Python 2.6, 2.7, 3.3, 3.4 and 3.5, so you'll need a way to switch between them. Once activated, it prefixes the PATH environment variable with ~/.pyenv/shims, where there are special files matching the Python commands (python, pip). These are not copies of the Python-shipped commands; they are special scripts that decide on the fly which version of Python to run based on the $\text{PYENV}_{\text{VERSION}}$ environment variable, or the .python-version file, or the ~/.pyenv/version file. pyenv also makes the process of downloading and installing multiple Python versions easier, using the command pyenv install.

5.2.1 Installation of pyenv and extensions INSTALL

We won't be installing pyenv through zypper since zypper doesn't have it unless you add someone's personal repo (which I am unwilling to do). Instead, we'll be installing it through cloning a git repo. Since pyenv is just a bunch of shell scripts, we'll be alright.

```
# Taken from https://www.reddit.com/r/openSUSE/comments/70ozge/using_multiple_python_ve

git clone https://github.com/pyenv/pyenv.git ~/.pyenv
echo 'export PYENV_ROOT="$HOME/.pyenv"' >> ~/.bashrc
echo 'export PATH="$PYENV_ROOT/bin:$PATH"' >> ~/.bashrc
echo -e 'if command -v pyenv 1>/dev/null 2>&1; then\n  eval "$(pyenv init -)"\nfi' >> 
```

Install the missing headers needed by Python modules

```
readline-devel sqlite3-devel libbz2-devel zlib-devel libopenssl-devel
```

Install virtualvenv

```
python3-virtualenv
```

```
@test "Check if pyenv has installed successfully" {
    command -v pyenv
}
```

5.2.2 Installing different versions of python

Installing new Python versions is very straightforward. All Python versions are installed in the versions directory under the pyenv root.

```
pyenv install 3.6.0
pyenv install 2.7.13
```

Figure 6: Install CPython 3.6.0 and CPython 2.7.13.

5.2.3 virtualvenv setup

With virtualenv all your virtualenvs are kept on a same directory and your projects' code on another. My setup is:

```
# All virtualenvs will be on...
# export WORKON_HOME=~/.ve
mkdir -p ~/.ve

# All projects will be on...
# export PROJECT_HOME=~/Projects
mkdir -p ~/Projects

# The -p flag is in case these folders have been created earlier - without it, mkdir re
```

It's necessary to configure the shell to initialize pyenv when you start a terminal session. Put the lines bellow on your ~/.bashrc file:

```
export PATH="~/.pyenv/bin/:$PATH"

export WORKON_HOME=~/.ve
export PROJECT_HOME=~/Projects
if which pyenv > /dev/null; then eval "$(pyenv init -)"; fi
```

5.2.4 Resist the temptation to contaminate your global Python install

I frequently use programs written in Python. I like them to be available in all sessions without activate any virtualenv.

However I don't like to mess with the global Python installation to avoid library conflict issues.

Another thing that I don't like is installing Jupyter/iPython on each of my projects' virtualenvs.

I like to have only one install of Jupyter Notebook , one of iPython Console for Python3, one of iPython Console for Python2, and other tools like youtube-dl, rename, gnucash-to-beancount, rows, s3cmd, fabric, mercurial, etc.

```
pyenv virtualenv 3.6.0 jupyter3
pyenv virtualenv 3.6.0 tools3
pyenv virtualenv 2.7.13 ipython2
pyenv virtualenv 2.7.13 tools2
```

Jupyter supports many kernels. This allows a single Jupyter install to create notebooks for Python2, Python3, R, Bash and many other languages. At this time I only want to support Python2 and Python3.

5.2.4.1 Installing jupyter under jupyter3

```
pyenv activate jupyter3
pip install jupyter
python -m ipykernel install --user
pyenv deactivate
```

5.2.4.2 Installing ipython under ipython2

```
pyenv activate ipython2
pip install ipykernel
python -m ipykernel install --user
pyenv deactivate
```

Note that when I install Jupyter on Python3 it will by default install iPython and the Kernel too. For Python2 I only need to install iPython and the Kernel. I'll explain this better bellow.

5.2.4.3 Tools which run on Python 3

```
pyenv activate tools3
pip install youtube-dl gnucash-to-beancount rows
pyenv deactivate
```

5.2.4.4 Tools that only run on Python 2

```
pyenv activate tools2
pip install rename s3cmd fabric mercurial
pyenv deactivate
```

5.2.4.5 Final Step Finally, it's time to make all Python versions and special virtualenvs work with each other.

```
pyenv global 3.6.0 2.7.13 jupyter3 ipython2 tools3 tools2
```

The above command establishes the PATH priority so scripts can be accessed in the right order without activating any virtualenv.

5.2.5 How to use Jupyter and iPython with my projects?

This was the main motivation to write this guide.

Both Notebook and Console were part of the iPython project, which, as the name suggests, were only about Python. But the Notebook evolution enabled it to become language agnostic, so developers decided to split the project in 2: Jupyter and iPython

Now Jupyter contains Notebook, while iPython contains Console and the Python Kernel which Jupyter uses to execute Python code.

I used to use an old iPython version and during a clumsy upgrade Jupyter stopped detecting the active virtualenv, so I couldn't import its installed libraries.

Actually, Jupyter does not detect the active virtualenv: it's the iPython instance which Jupyter initializes. The problem then is that iPython's virtualenv detection code only runs in the interactive shell mode, but not in the kernel mode. Besides that the detection code only works properly if the active virtualenv's Python version and the Python version running iPython are the same.

The solution is to customize iPython's startup process. For that we need to create an iPython profile and install a magic script I wrote to do the trick:

```
ipython profile create
curl -L http://hbn.link/hb-ipython-startup-script > ~/.ipython/profile_default/startup/
```

With this, no matter the mode iPython starts, the virtualenv's site-packages will be available in the PYTHONPATH.

Back to our proj3, after activating its virtualenv running workon proj3, you can simply execute ipython to run the interactive mode, or jupyter notebook to get all the fun.

5.3 Pylint

6 Bash

6.1 bats-core

bats-core is a unit test library for

```
bats
```

```
git clone https://github.com/bats-core/bats-core.git
cd bats-core
sudo ./install.sh /usr/local
```

7 Haskell

7.1 Spacemacs Layer SPACEMACS

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; haskell layer configuration                                        ;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
(haskell :variables                                                   ;;
         haskell-completion-backend 'intero)                          ;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

8 Markup Languages

8.1 csv

Probably not markup but close enough

8.1.1 Spacemacs Layer SPACEMACS

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; csv layer configuration      ;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
csv                               ;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

8.2 html

8.2.1 Spacemacs Layer SPACEMACS

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; html layer configuration    ;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
html                              ;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

8.3 markdown

8.3.1 Spacemacs Layer SPACEMACS

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; markdown layer configuration   ;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
markdown                           ;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

8.4 yaml

8.4.1 Spacemacs Layer SPACEMACS

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; yaml layer configuration   ;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
yaml                            ;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

8.5 org

8.5.1 Spacemacs Layer SPACEMACS

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

```
;; org layer configuration  ;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
org                          ;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

8.6 asciidoc

8.6.1 Spacemacs Layer SPACEMACS

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; asciidoc layer configuration   ;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
asciidoc                           ;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

8.7 dot & graphviz

8.7.1 Spacemacs Layer SPACEMACS

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; graphviz layer configuration   ;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
graphviz                           ;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

9 Javascript

9.0.1 Spacemacs Layer SPACEMACS

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; javascript layer configuration ;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
javascript                         ;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

10 Emacs-lisp

10.0.1 Spacemacs Layer SPACEMACS

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; emacs-lisp layer configuration   ;;
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
emacs-lisp                              ;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

11 C & C++

11.0.1 Spacemacs Layer SPACEMACS

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; c & C++ layer configuration    ;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
c-c++                                   ;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

12 Spacemacs

12.1 iBuffer

12.1.1 Spacemacs Layer SPACEMACS

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; iBuffer configuration                    ;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
(ibuffer :variables                         ;;
         ibuffer-group-buffers-by 'mode) ;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

13 gpg.conf

This is an implementation of the Riseup OpenPGP Best Practices `https://help.riseup.net/en/security/message-security/openpgp/best-practices`

13.1 default key

The default key to sign with. If this option is not used, the default key is the first key found in the secret keyring

```
default-key 0x18F3685C0022BFF3
```

13.2 behavior

13.2.1 Disable inclusion of the version string in ASCII armored output

```
no-emit-version
```

13.2.2 Disable comment string in clear text signatures and ASCII armored messages

```
no-comments
```

13.2.3 Display long key IDs

```
keyid-format 0xlong
```

13.2.4 List all keys (or the specified ones) along with their fingerprints

```
with-fingerprint
```

13.2.5 Display the calculated validity of user IDs during key listings

```
list-options show-uid-validity
verify-options show-uid-validity
```

13.2.6 Try to use the GnuPG-Agent. With this option, GnuPG first tries to connect to the agent before it asks for a passphrase.

```
use-agent
charset utf-8
fixed-list-mode
```

13.3 keyserver

This is the server that –recv-keys, –send-keys, and –search-keys will communicate with to receive keys from, send keys to, and search for keys on

```
#keyserver hkps://hkps.pool.sks-keyservers.net
keyserver pgp.mit.edu
```

Provide a certificate store to override the system default Get this from `https://sks-keyservers.net/sks-keyservers.netCA.pem`

```
#keyserver-options ca-cert-file=/usr/local/etc/ssl/certs/hkps.pool.sks-keyservers.net.p
```

Set the proxy to use for HTTP and HKP keyservers - default to the standard local Tor socks proxy It is encouraged to use Tor for improved anonymity. Preferrably use either a dedicated SOCKSPort for GnuPG and/or enable IsolateDestPort and IsolateDestAddr I run my tor socks proxy in a container, see .dockerfunc and github.com/jfrazelle/dockerfiles

```
#keyserver-options http-proxy=socks5-hostname://torproxy:9050
```

Don't leak DNS, see `https://trac.torproject.org/projects/tor/ticket/2846`

```
#keyserver-options no-try-dns-srv
```

When using –refresh-keys, if the key in question has a preferred keyserver URL, then disable use of that preferred keyserver to refresh the key from

```
keyserver-options no-honor-keyserver-url
```

When searching for a key with –search-keys, include keys that are marked on the keyserver as revoked

```
keyserver-options include-revoked
```

13.4 algorithm and ciphers

list of personal digest preferences. When multiple digests are supported by all recipients, choose the strongest one

```
personal-cipher-preferences AES256 AES192 AES CAST5
```

list of personal digest preferences. When multiple ciphers are supported by all recipients, choose the strongest one

```
personal-digest-preferences SHA512 SHA384 SHA256 SHA224
```

message digest algorithm used when signing a key

```
cert-digest-algo SHA512
s2k-cipher-algo AES256
s2k-digest-algo SHA512
```

This preference list is used for new keys and becomes the default for "setpref" in the edit menu

```
default-preference-list SHA512 SHA384 SHA256 SHA224 AES256 AES192 AES CAST5 ZLIB BZIP2
```