```
#!/bin/bash
# shellcheck source=kz_common.sh
##############################################################################
# Build development environment.
#
# Written in 2023 by Karel Zimmer <info@karelzimmer.nl>, Creative Commons
# Public Domain Dedication <https://creativecommons.org/publicdomain/zero/1.0>.
##############################################################################

set -o errexit
program_path=$(cd "$(dirname "$(realpath "$0")")" && pwd)
source "$program_path"/kz_common.sh


##############################################################################
# Constants
##############################################################################

declare program_name='kz-getdev'
declare program_desc
program_desc=$(gettext 'Build development environment')
declare display_name=${program_name/kz-/kz }

declare usage
usage=$(eval_gettext "Usage: \$display_name \$options_usage")
declare help
help="$(eval_gettext "Usage: \$display_name [OPTION...]")

$program_desc.

$(gettext 'Options:')
$options_help"


##############################################################################
# Variables
##############################################################################


##############################################################################
# Functions
##############################################################################

function check_input {
    local -i getopt_rc=0
    local parsed=''

    parsed=$(
        getopt  --alternative                       \
                --options       "$options_short"    \
                --longoptions   "$options_long"     \
                --name          "$display_name"     \
                --              "$@"
        ) || getopt_rc=$?
    if [[ $getopt_rc -ne $ok ]]; then
        info "$usage_line"
        exit $error
    fi
    eval set -- "$parsed"
    kz_common.process_options "$@"

    while true; do
        case $1 in
            --)
```

```
                shift
                break
                ;;
            *)
                shift
                ;;
        esac
    done

    if [[ "$*" ]]; then
        printf  "$display_name: $*: %s\n$usage_line\n" \
                "$(gettext 'arguments are not allowed')"
        exit $error
    fi
}


function process_input {
    if groups "$USER" | grep --quiet --regexp='sudo'; then
        sudo true
    else
        printf  '%s\n' "$(gettext 'Already performed by the administrator.')"
        term_script
    fi
    info "* ${bold}Check${normal} ($display_name)"
    check_dependencies
    info "\n* ${bold}Build${normal} ($display_name)"
    download_website
    pull_repos
}


function check_dependencies {
    local escape='gettext'

    printf '%s\n' "$(gettext '* Check dependencies...')"
    # Install ghostscript for kz-build <man-pag>.pdf (ps2pdf) and remove known
    # lines from output.
    check_dpkgd_snapd
    sudo    apt-get                                 \
            install                                 \
            --yes                                   \
            curl                                    \
            fakeroot                                \
            $escape                                 \
            ghostscript                             \
            git                                     \
            jq                                      \
            lftp                                    \
            nmap                                    \
            pycodestyle                             \
            python3-pycodestyle                     \
            python3-autopep8                        \
            python3-pip                             \
            python-is-python3                       |
            sed '/is reeds de nieuwste versie /d'   |
            sed '/is already the newest version /d'
    sudo ln --force --relative --symbolic /usr/bin/pycodestyle /usr/bin/pep8
    sudo ln --force --relative --symbolic /usr/bin/pip3 /usr/bin/pip
    # Debian packages are old, snap is newer and remove known lines from
    # output.
    check_dpkgd_snapd
    sudo snap install shellcheck |& sed '/is already installed/d'
    check_dpkgd_snapd
```

```
    sudo snap install --classic code |& sed '/is already installed/d'
}


function check_dpkgd_snapd {
    local -i dpkg_wait=10
    local text
    text=$(eval_gettext \
            "Wait \${dpkg_wait}s for another package manager to finish...")

    if find /snap/core/*/var/cache/debconf/config.dat &> /dev/null; then
        # System with snaps.
        while sudo  fuser                                               \
                    /var/{lib/{dpkg,apt/lists},cache/apt/archives}/lock \
                    /var/cache/debconf/config.dat                       \
                    /snap/core/*/var/cache/debconf/config.dat           \
                    &> /dev/null; do
            printf '%s\n' "$text"
            sleep $dpkg_wait
        done
    else
        # System without snaps.
        while sudo  fuser                                               \
                    /var/{lib/{dpkg,apt/lists},cache/apt/archives}/lock \
                    /var/cache/debconf/config.dat                       \
                    &> /dev/null; do
            printf '%s\n' "$text"
            sleep $dpkg_wait
        done
    fi
}


function download_website {
    local ftp_set='set ssl:verify-certificate no'
    local ftp_from=/httpdocs
    local ftp_to=$HOME/kz-uploads/dist
    local ftp_opts='--delete --verbose'
    local ftp_exclude='--exclude icaclient_20.04.0.21_amd64.deb'
    local ftp_cmd="mirror $ftp_exclude $ftp_opts $ftp_from $ftp_to; exit"
    local ftp_host=server106.hosting2go.nl
    local ftp_user=kzimmer
    local ftp_login=$HOME/.kz-$ftp_host

    printf '%s\n' '* Download website (lftp)...'
    if ! [[ -f $ftp_login ]]; then
        read -rsp "$(gettext 'Password for') ftp://$ftp_host': " < /dev/tty
        printf '%s\n' "$REPLY" > "$ftp_login"
        printf '\n'
        chmod 'u=rw,g=,o=' "$ftp_login"
    fi
    if ! lftp   --user "$ftp_user"                  \
                --password "$(cat "$ftp_login")"    \
                -e "$ftp_set; $ftp_cmd"             \
                "$ftp_host"; then
        rm  "$ftp_login"
        printf '%s\n' "$(gettext 'Website download failed.')"
        return  1
    fi
}


function pull_repos {
    local bin_repo=/home/"$USER"/bin
```

```
    git config --global user.name 'Karel Zimmer'
    git config --global user.email 'karel.zimmer@gmail.com'
    git config --global pull.ff only
    git config --global credential.helper store

    if [[ -d "$bin_repo" ]]; then
        cd "$bin_repo" || exit 1
        # Remove known line from output.
        git pull | sed '/Already up to date./d' | sed '/Already up-to-date./d'
    else
        git clone https://github.com/karelzimmer/bin.git "$bin_repo"
    fi
    printf '%s\n' '* Pull repos (gitpull)...'
    "$bin_repo"/gitpull
    printf '%s\n' '* Status repos (gitstat)...'
    /home/"$USER"/bin/gitstat
}


function term_script {
    exit $ok
}


################################################################################
# Script
################################################################################

function main {
    kz_common.init_script "$@"
    check_input "$@"
    process_input
    term_script
}

main "$@"
```