#FYD500 - Homework V The first number of assignments for FYD500, an introduction to linux.

##Advanced shell variables

Write a script that does the following, *Display the name of the script being executed* Display the first, third and tenth argument given to the script. * Display total number of arguments * If there were more than three positional arguments passed to script, use shift to move all the values 3 places to the left. * Print the remaining parameters left in the positional parameters. * Print the number of arguments.

```
#cat advanced_shell_var.sh
-------------------------------------------
#!/bin/bash

POS=()
_file=$(echo ${0} | awk -F '/' '{print $NF}')
clear
printf "The script has the name %s\n" $_file

echo "These are all the arguments: "
while [[ $# -gt 0 ]]
do
    printf "%s\n" $1
    POS+=("$1")
    shift
done

set "${POS[@]}"

echo "==================="
printf "Argument 1 =$1\n\
    Argument 1 =$2\n\
    Argument 2 =$3\n\
    Argument 10 =${10}\n"

printf "The total number of arguments is: %d\n" $#

if [[ $# -gt 3 ]];then
    shift;shift;shift
fi

printf "Argument 1 =$1\n\
    Argument 1 =$2\n\
    Argument 1 =$3\n"
```

```
while [[ $# -gt 0 ]]
do
    printf "\nCurrent argument is: %s" $1
    shift
done
```

##Repetition constructs Reconsider the exercise from 5.2. Rewrite the script with repetition construct.

```
#!/usr/bin/env bash

_msg_help="The script will print home path [-o], current shell[-s], free memory[-m], swap sp
_msg_empty="Please supply an input argument, see --help for all alternatives."
_msg_error="Error, please see --help for available flags."

# Unset parameters
unset h s m p help

# Assign mem and swp to free memory and swap
_mem=$(free -hm | awk  'NR==2 { print $3}')
_swp=$(free -hm | awk  'NR==3 { print $3}')
_home=$(pwd)
_csh=${SHELL}

# Get opts does not support long options i.e. --help

for arg in "$@"; do
    shift
    case "$arg" in
        "--home")   set -- "$@" "-o" ;;
        "--shell")  set -- "$@" "-s" ;;
        "--mem")    set -- "$@" "-m" ;;
        "--swap")   set -- "$@" "-w" ;;
        "--help")   set -- "$@" "-h" ;;
        *)          set -- "$@" "$arg"
    esac
done

# Check
_flags=1
while getopts "osmwh" opt
do
    case "$opt" in
        "o") printf "%-6s\t%-8s\n" "HOME:"  "${_home}";;
        "s") printf "%-6s\t%-8s\n" "SHELL:" "${_csh}";;
        "m") printf "%-6s\t%-8s\n" "MEM:"   "${_mem}";;
        "w") printf "%-6s\t%-8s\n" "SWAP:"  "${_swp}";;
```

```
        "h") clear; echo ${_msg_help};;
        "") clear;  printf "%s" "${_msg_empty}";exit 1;;
        "?") clear; printf "%s" "${_msg_error}";exit 2;;
    esac
done
shift $(expr $_flags - 1)
exit 0
```

Can you explain why it is so important to put the variable in between double qoutes?

```
#!/usr/bin/env bash
ARCHIVENR=$(date +%Y%m%d)
DESTDIR="$PWD/archive-$ARCHIVENR"
mkdir "$DESTDIR"

find "$PWD" -type f -a -mtime +5 | while read -d $'\000' file
do
    gzip "$file"; mv "$file".gz "$DESTDIR"
    echo "$file archived"
done
```

Single quotes would interpet the commands as text, ie. echo '$(pwd)' would yield $(pwd). Without any quotes, the complete command wouldn"t register.

Write a similar similar to the one in 9.5.1 but think of a way of quiting after the user has executed the loop three times.

```
#!/bin/bash
_COUNTER=0
FORTUNE=/usr/games/fortune
while true; do
    _COUNTER=$((\_COUNTER+1))
    echo "On which topic do you want advice?"
    echo "1. politics"
    echo "2. startrek"
    echo "3. kernelnewbies"
    echo "4. sports"
    echo "5. bofh-excuses"
    echo "6. magic"
    echo "7. love"
    echo "8. literature"
    echo "9. drugs"
    echo "10. education"
    case $choice in
        1)
            $FORTUNE politics
        ;;
```

```
        2)
            $FORTUNE startrek
        ;;
        3)
            $FORTUNE kernelnewbies
        ;;
        4)
            echo "Sports are a waste of time, energy and money."
            echo "Go back to your keyboard."
            echo -e "\t\t\t\t -- \"Unhealthy is my middle name\" Soggie."
        ;;
        5)
            $FORTUNE bofh-excuses
        ;;
        6)
            $FORTUNE magic
        ;;
        7)
            $FORTUNE love
        ;;
        8)
            $FORTUNE literature
        ;;
        9)
            $FORTUNE drugs
        ;;
        10)
            $FORTUNE education
        ;;
        0)
            echo "OK, see you!"
            break
        ;;
        *)
            echo "That is not a valid choice, try a number from 0 to 10."
        ;;
    esac
if [[ $_COUNTER == 3 ]]
then
    break
fi
echo $_COUNTER
done
```

##Functions

Add a funtion to your .bashrc that automates the generation of man pages in PDF format.

```
# Print man pages in pdf format
    function printman() {
        if [[ -n $1 ]]
            then
                _MANUAL=$1
        else
            _MANUAL=$(man -k . |  dmenu -l 15 | awk '{print $1}')
                fi
                man -Tpdf $_MANUAL | zathura -
    }
```