# lusc Bash version

```bash
#!/bin/bash
# LUSC - Linux UEFI STUB Creator
# 2024 by Lennart Martens - monkeynator78@gmail.com - https://github.com/lennart1978/LUSC -
# Automatically generate UEFI boot entries

# Color Codes
BLUE=$(tput setaf 4)
GREEN=$(tput setaf 2)
RED=$(tput setaf 1)
RESET=$(tput sgr0)

# Display usage information
usage() {
    cat << EOF
Usage: $(basename "$0")
This is a simple interactive tool to automatically generate UEFI boot entries.
It generates efibootmgr commands and exports them to a small executable.
No changes will be written to disk before confirmation.
The EFI partition must be mountet to /boot and the kernel -and initramfs image must be located at the root of it !
Some UEFI systems don't allow to create more than one EFI STUB entry.
Unfortunately efibootmgr is not able to change EFI entries. You always have to delete/overwrite entries to make changes happen.
Please don't use this Bash script when you don't exactly know what you are doing here and what EFI STUB means.
You can get some great info at : https://wiki.archlinux.org/title/EFISTUB

And now good luck with EFI STUB booting.
L.Martens

Options:
    -h, --help      Display this help message
EOF
}

# Check if the script is running with root privileges
if [ "$UID" -ne 0 ]; then
    echo "${RED}This script must be run with root privileges !${RESET}"
    echo "type: sudo lusc -h for usage and more info."
    exit 1
fi

# Parse command-line arguments
while [[ $# -gt 0 ]]; do
    case "$1" in
        -h | --help)
            usage
            exit 0
            ;;
        *)
            echo "Unknown option: $1"
            usage
            exit 1
            ;;
    esac
    shift
done

# Prompt user to continue
echo "${BLUE}Welcome to LUSC - A Linux UEFI STUB Creator"
echo "-----------------------------------------"
echo "-----------------------------------------${RESET}"
read -r -p "Start creating UEFI boot entries ? (y/N) " choice
choice=$(echo "$choice" | tr '[:upper:]' '[:lower:]')
if [[ "$choice" != "y" ]]; then
    echo "Goodbye.Exiting..."
    exit 0
fi


# Prompt user to specify EFI partition
read -r -p "Please specify EFI partition (e.g., /dev/nvme0n1p1): " efi_partition

# Check if the EFI partition exists
if ! blkid | grep -q "$efi_partition"; then
    echo "${RED}Error: EFI partition '$efi_partition' not found !${RESET}"
    exit 1
fi

# Extract disk and partition number
efi_disk=$(echo "$efi_partition" | sed -E 's/p?[0-9]+$//')
efi_part_num=$(echo "$efi_partition" | grep -o '[0-9]*$')

# Prompt user to specify the label for the boot entry
read -r -p "Please specify the label for the boot entry (e.g., Arch Linux): " boot_label

# Detect partitions
efi_uuid=$(blkid -o value -s UUID "$efi_partition")
root_uuid=$(blkid -o value -s UUID "$(findmnt -no SOURCE /)")

# Check if swap partition exists
swap_uuid=$(blkid -o value -s UUID "$(findmnt -no SOURCE /swap)")
if [[ -z "$swap_uuid" ]]; then
    echo "${GREEN}No swap partition detected. Assuming Zswap is used.${RESET}"
fi

# Default kernel parameters
default_params="root=UUID=$root_uuid rw"
if [[ -n "$swap_uuid" ]]; then
    default_params="$default_params resume=UUID=$swap_uuid"
fi

# Prompt user to specify additional kernel parameters
echo "Current kernel parameters: $default_params"
echo "${GREEN}initrd and initrd-fallback will be added automatically !${RESET}"
echo "For example additional kernel parameters could be: quiet splash rootfstype=ext4 hostname=my-computer nohibernate noresume vm_debug=- ..."
read -r -p "Add additional kernel parameters (or press Enter to keep current): " extra_params

# Combine default and additional parameters
```

```bash
if [[ -n "$extra_params" ]]; then
    kernel_params="$default_params $extra_params"
else
    kernel_params="$default_params"
fi

# initramdisks with "\" !
initramdisk="\initramfs-linux.img"
initfallback="\initramfs-linux-fallback.img"

# Compose the command strings
linux_cmd="efibootmgr --create --disk $efi_disk --part $efi_part_num --label \"$boot_label\" --loader /vmlinuz-linux --unicode \"$kernel_params initrc
fallback_cmd="efibootmgr --create --disk $efi_disk --part $efi_part_num --label \"$boot_label (Fallback)\" --loader /vmlinuz-linux --unicode \"$kernel

# Print the commands for user confirmation
echo "Detected partitions:"
echo "EFI: $efi_partition ($efi_uuid)"
echo "Root: $(findmnt -no SOURCE /) ($root_uuid)"
if [[ -n "$swap_uuid" ]]; then
    echo "Swap: $(findmnt -no SOURCE /swap) ($swap_uuid)"
    resume_option="resume=UUID=$swap_uuid"
else
    resume_option=""
fi

echo

echo "Composed commands:"
echo "$linux_cmd"
echo "$fallback_cmd"


# Prompt user to write or execute commands
read -r -p "Create executable only, create and execute (sets UEFI boot entries), or abort? (c/ce/a) " action
action=$(echo "$action" | tr '[:upper:]' '[:lower:]')
case "$action" in
    c)
        # Write commands to file
        script_file="uefi_stub_gen_$(date "+%d-%-m-%Y--%H:%M")"
        {
            echo "#!/bin/bash"
            echo "# Generated UEFI boot entries by LUSC"
            echo "$fallback_cmd"
            echo "$linux_cmd"
            echo "exit 0"
            echo "# See 'man efibootmgr' for more information"
        } > "$script_file"
        chmod +x "$script_file"
        echo "Commands written to file: $script_file"
        ;;
    ce)
        # Write commands to file and execute
        script_file="uefi_stub_gen_$(date "+%d-%-m-%Y--%H:%M")"
        {
            echo "#!/bin/bash"
            echo "# Generated UEFI boot entries by LUSC"
            echo "$fallback_cmd"
            echo "$linux_cmd"
            echo "exit 0"
            echo "# See 'man efibootmgr' for more information"
        } > "$script_file"
        chmod +x "$script_file"
        echo "Commands written to file: $script_file"
        echo "Executing commands..."
        "./$script_file"
        echo "Changes written. Power off and restart (${RED}don't reboot !${RESET})."
        ;;
    a)
        echo "Aborted. No changes made."
        ;;
    *)
        echo "${RED}Invalid choice ! Aborting.${RESET}"
        ;;
esac

exit 0
```