

nzk-scripts

- [nzk-app-add](#)
- [install log fail](#)
- [py scripts from other repo's that i use \(need to add to my repo\)](#)

nzk-app-add

```
#!/bin/bash

#nzk-app-add
#i use kali linux
# Check for at least one argument
if [ "$#" -lt 1 ]; then
    echo "Usage: nzk-app-add <executable> [--icon <iconpath>]"
    exit 1
fi

EXECUTABLE=$(realpath "$1")
APP_NAME=$(basename "$EXECUTABLE")
ICON_PATH=""

# Check for optional icon argument
if [ "$2" == "--icon" ] && [ -n "$3" ]; then
    ICON_PATH=$(realpath "$3")
fi

# Create .desktop file
DESKTOP_ENTRY="$HOME/.local/share/applications/$APP_NAME.desktop"
echo "[Desktop Entry]" > "$DESKTOP_ENTRY"
echo "Type=Application" >> "$DESKTOP_ENTRY"
echo "Name=$APP_NAME" >> "$DESKTOP_ENTRY"
echo "Exec=$EXECUTABLE" >> "$DESKTOP_ENTRY"
echo "Icon=$ICON_PATH" >> "$DESKTOP_ENTRY"
echo "Terminal=false" >> "$DESKTOP_ENTRY"

# Make the .desktop file executable
chmod +x "$DESKTOP_ENTRY"

echo "$APP_NAME added to applications menu."
```

install log fail

```
└─(kali㉿kali)-[~]  
└─$ cd nzk-apps
```

```
└─(kali㉿kali)-[~/nzk-apps]  
└─$ ls
```

```
bash install-NZK-Scripts.sh LICENSE nzk-code README.md
```

```
└─(kali㉿kali)-[~/nzk-apps]  
└─$ bash ./install-NZK-Scripts.sh
```

Are you NaruZKurai? (y/n)

If you aren't and you say yes, things will break.

So if you are you NaruZKurai, v1 will be installed and uses /home/kali as default user path

Please enter your answer: n

Cool, let's get started.

nzk-code3 copied successfully.

cp: cannot create regular file '/usr/local/bin/nzk-nature': Permission denied

Failed to copy nzk-nature. Ensure the file exists and try again.

Installation complete.

```
└─(kali㉿kali)-[~/nzk-apps]  
└─$ cat ./bash/ nzk-nature
```

cat: ./bash/: Is a directory

cat: nzk-nature: No such file or directory

```
└─(kali㉿kali)-[~/nzk-apps]  
└─$ cat ./bash/nzk-nature
```

```
#!/bin/bash
```

```
#nzk-nature
```

```
# Define associative arrays for natures and their properties
```

```
CONFIG_FILE="$~/.config/nzk-apps/settings.conf"
```

```
declare -A nature_stats=(
```

```
  [0]="— —" [1]="Attack Defense" [2]="Attack Speed" [3]="Attack Sp. Attack" [4]="Attack Sp.  
Defense"
```

```
  [5]="Defense Attack" [6]="— —" [7]="Defense Speed" [8]="Defense Sp. Attack" [9]="Defense  
Sp. Defense"
```

```
  [10]="Speed Attack" [11]="Speed Defense" [12]="— —" [13]="Speed Sp. Attack" [14]="Speed  
Sp. Defense"
```

```
  [15]="Sp. Attack Attack" [16]="Sp. Attack Defense" [17]="Sp. Attack Speed" [18]="— —"  
[19]="Sp. Attack Sp. Defense"
```

```
  [20]="Sp. Defense Attack" [21]="Sp. Defense Defense" [22]="Sp. Defense Speed" [23]="Sp.
```

```

Defense Sp. Attack" [24]="— —"
)
declare -A nature_flavors_english=(
    [0]="— —" [1]="Spicy Sour" [2]="Spicy Sweet" [3]="Spicy Dry" [4]="Spicy Bitter"
    [5]="Sour Spicy" [6]="— —" [7]="Sour Sweet" [8]="Sour Dry" [9]="Sour Bitter"
    [10]="Sweet Spicy" [11]="Sweet Sour" [12]="— —" [13]="Sweet Dry" [14]="Sweet Bitter"
    [15]="Dry Spicy" [16]="Dry Sour" [17]="Dry Sweet" [18]="— —" [19]="Dry Bitter"
    [20]="Bitter Spicy" [21]="Bitter Sour" [22]="Bitter Sweet" [23]="Bitter Dry" [24]="— —"
)
declare -A nature_flavors_japanese=(
    [0]="— —" [1]="辛い 酸味" [2]="辛い 甘味" [3]="辛い 乾味" [4]="辛い 苦味"
    [5]="酸味 辛い" [6]="— —" [7]="酸味 甘味" [8]="酸味 乾味" [9]="酸味 苦味"
    [10]="甘味 辛い" [11]="甘味 酸味" [12]="— —" [13]="甘味 乾味" [14]="甘味 苦味"
    [15]="乾味 辛い" [16]="乾味 酸味" [17]="乾味 甘味" [18]="— —" [19]="乾味 苦味"
    [20]="苦味 辛い" [21]="苦味 酸味" [22]="苦味 甘味" [23]="苦味 乾味" [24]="— —"
)
declare -A nature_flavors_romanji=(
    [0]="— —" [1]="Karaī Suppai" [2]="Karaī Amai" [3]="Karaī Noma" [4]="Karaī Nigai"
    [5]="Suppai Karaī" [6]="— —" [7]="Suppai Amai" [8]="Suppai Noma" [9]="Suppai Nigai"
    [10]="Amai Karaī" [11]="Amai Suppai" [12]="— —" [13]="Amai Noma" [14]="Amai Nigai"
    [15]="Noma Karaī" [16]="Noma Suppai" [17]="Noma Amai" [18]="— —" [19]="Noma Nigai"
    [20]="Nigai Karaī" [21]="Nigai Suppai" [22]="Nigai Amai" [23]="Nigai Noma" [24]="— —"
)
# Define associative arrays for English, Japanese, and Romanji nature names
declare -A english_natures=(
    [0]="Hardy" [1]="Lonely" [2]="Brave" [3]="Adamant" [4]="Naughty"
    [5]="Bold" [6]="Docile" [7]="Relaxed" [8]="Impish" [9]="Lax"
    [10]="Timid" [11]="Hasty" [12]="Serious" [13]="Jolly" [14]="Naive"
    [15]="Modest" [16]="Mild" [17]="Quiet" [18]="Bashful" [19]="Rash"
    [20]="Calm" [21]="Gentle" [22]="Sassy" [23]="Careful" [24]="Quirky"
)
declare -A japanese_natures=(
    [0]="元気" [1]="寂しい" [2]="勇敢" [3]="不屈" [4]="獰猛"
    [5]="勇敢" [6]="温順" [7]="寛容" [8]="皮肉" [9]="怠慢"
    [10]="臆病" [11]="急ぎ" [12]="真面目" [13]="陽気" [14]="天真"
    [15]="謙虚" [16]="穏当" [17]="静か" [18]="臆い" [19]="冒険"
    [20]="冷静" [21]="優しい" [22]="陽気" [23]="慎重" [24]="変人"
)
declare -A romanji_natures=(
    [0]="Gannbariya" [1]="Samishigari" [2]="Yuukan" [3]="Ijippari" [4]="Yancha"
    [5]="Zubutoi" [6]="Sunao" [7]="Nonnki" [8]="Wanpaku" [9]="Noutennki"
    [10]="Okubiyoi" [11]="Sekkachi" [12]="Majime" [13]="Youki" [14]="Mujiyaki"
    [15]="Hikaeme" [16]="Ottori" [17]="Reisei" [18]="Tereya" [19]="Ukkariya"
    [20]="Odayaka" [21]="Otonashii" [22]="Namaiki" [23]="Shinnchyou" [24]="Kimagure"
)
# Function to find the index of a nature name, case-insensitive

```

```

find_nature_index() {
    local nature_name_lower=$(echo "$1" | tr '[:upper:]' '[:lower:]')
    local -n arr=$2
    for i in "${!arr[@]}; do
        if [[ "$(echo "${arr[$i]}" | tr '[:upper:]' '[:lower:]')" == "$nature_name_lower" ]]; then
            echo $i
            return
        fi
    done
    echo "-1"
}

```

```

# Check for correct number of arguments
if [ $# -ne 2 ]; then
    echo "Usage: $0 <nature_name> [-e|-j|-r]"
    exit 1
fi

```

```

# Get the nature name and flag
nature_name="$1"
flag="$2"

```

```

# Main logic to get nature information based on flag and nature name
case $flag in
    -e)
        index=$(find_nature_index "$nature_name" english_natures)
        if [[ $index -ne -1 ]]; then
            echo "${japanese_natures[$index]}"
        else
            echo "Error: Nature '$nature_name' not found in English natures."
        fi
        ;;
    -j)
        index=$(find_nature_index "$nature_name" japanese_natures)
        if [[ $index -ne -1 ]]; then
            echo "${english_natures[$index]}"
        else
            echo "Error: Nature '$nature_name' not found in Japanese natures."
        fi
        ;;
    -r)
        index=$(find_nature_index "$nature_name" romanji_natures)
        if [[ $index -ne -1 ]]; then
            echo "${english_natures[$index]}"
        else
            echo "Error: Nature '$nature_name' not found in Romanji natures."
        fi
    *)
        echo "Error: Invalid flag '$flag'."
        exit 1
    ;;
esac

```

```
fi
;;
*)
    echo "Invalid flag: $flag. Use -e for English to Japanese, -j for Japanese to English, -r for
Romanji to English."
    echo "correct usage would be something like:"
    echo "nzk-nature "jolly" -e"
    exit 1
;;
esac
```

```
└─(kali㉿kali)-[~/nzk-apps]
└─$
```

py scripts from other repo's that i use (need to add to my repo)

https://github.com/Crypto-Cat/CTF/blob/main/pentesting/gen_nmap.py

```
#!/bin/python
#
# can be found on crypto cat's repo,
# https://github.com/Crypto-Cat/CTF/blob/main/pentesting/gen_nmap.py
#
# sudo apt-get install python3 masscan nmap
#
# This script will take in Masscan results and produce an output which can be fed into NMap

# Save your Masscan result as mscan.txt then run this script to produce nmap.txt then you can run:
# while read item; do sudo nmap -sV -sC -sS -sU $item; done < nmap.txt; rm mscan.txt nmap.txt

# If you want to export to .xml file you can use the following command and then later use this script to merge
files: https://github.com/sidaf/scripts/blob/master/nmap_merge.py
# while read item; do filename=$(echo $item | grep -o "\^[S*]"); sudo nmap -O -sV -sC -sS -sU $item -oX
$filename.xml; done < nmap.txt

import re
from socket import inet_aton
from os import path

regex = re.compile(r"Discovered open port (\d+)\V(udp|tcp) on (\d+\.\d+\.\d+\.\d+)", re.I)

ip_list = {}

with open(path.abspath('mscan.txt')) as f:
    lines = f.readlines()
```

```
for line in lines:
    port = regex.match(line).group(1)
    protocol = regex.match(line).group(2)
    ip = regex.match(line).group(3)

    # Add the IP to dictionary if it's not already
    try:
        ip_list[ip]
    except KeyError:
        ip_list[ip] = {}

    # Add protocol to dictionary if it's not already
    try:
        ip_list[ip][protocol]
    except KeyError:
        ip_list[ip][protocol] = []

    # Append the port to the list
    ip_list[ip][protocol].append(port)
```

```
with open(path.abspath('nmap.txt'), 'a') as f:
    sorted_ips = sorted(ip_list.items(), key=lambda item: inet_aton(item[0]))
    for ip, protocols in sorted_ips:
        udp_ports = ""
        tcp_ports = ""

        # Check to see if any UDP ports were found
        try:
            for port in protocols['udp']:
                udp_ports += port + ','
        except KeyError:
            pass

        # Check to see if any TCP ports were found
        try:
            for port in protocols['tcp']:
                tcp_ports += port + ','
        except KeyError:
            pass
```



```
# Print IP and ports to file ready for NMap scan
if udp_ports and tcp_ports:
    line = ip + ' -p U:' + udp_ports + 'T:' + tcp_ports
elif udp_ports:
    line = ip + ' -p U:' + udp_ports
elif tcp_ports:
    line = ip + ' -p T:' + tcp_ports
f.write(line + '\n')
```