```
#!/usr/bin/env python
# -*- coding: latin1 -*-

# ESLBench frontend script by Matheus Boy - Unicamp - Campinas State University

import os, sys
from datetime import *

# === AUXILIARY FUNCTIONS ===

# cleans up the dir structure in order to commit to version control
def clean():
    right_files = []
    for root, dirs, files in os.walk(os.getcwd()):
        if '.svn' in dirs:
            dirs.remove('.svn')
        for f in files:
            if f[-1] == '~':
                os.remove(os.path.join(root, f))
            elif f[-2] == '.' and (f[-1] == 'x' or f[-1] == 'o'):
                os.remove(os.path.join(root, f))

# general Makefile creator
def makefile(proc, nproc, sw):
    return "#FILE GENERATED AUTOMATICALLY - DO NOT EDIT\nexport PROCESSOR := " \
            + proc + "\nexport NUMPROCESSORS := " + nproc + \
            "\nexport SOFTWARE := " + sw + "\nexport CROSS := " + proc + \
            "-elf-gcc\nexport PLATFORM := $(PROCESSOR).$(NUMPROCESSORS)\n\n" + \
            "include Makefile.conf\ninclude Makefile.rules\n"

# prints usage
def help():
    print "Usage: ./ESLBench --build -p=<processor> -n=<number_of_cores> -s=<software>"
    print "       ./ESLBench --run -p=<processor> -n=<number_of_cores> -s=<software>"
    print "       ./ESLBench --run -p=<processor> -n=<number_of_cores> -s=<software> --nobuild"
    print "       ./ESLBench --clean"
    print "       ./ESLBench --help"

# rundir Makefile creator
def run_make(path, proc, nproc, sw):
    make = "run:\n\t./" + proc + "." + nproc + ".x --load=" + sw + "." + proc + ".x " + nproc +"\n"
    f = open(path + "/Makefile", "w")
    f.write(make)
    f.close()

# rundir creation
def build(tracker):
    path = ""
    try:
        # walks the platform tracker make tree
        for i in tracker[0]:
            for j in tracker[1]:
                for k in tracker[2]:
                    os.system("rm Makefile")
                    # creates general Makefile
                    f = open("Makefile", "w")
                    f.write(makefile(i, j, k))
                    f.close()
                    # makes the platform
                    os.system("make clean all")
                    path = "rundir/" + i + "." + j + "." + k
                    print "Creating rundir for " + path[7:] + "..."
                    # creates rundir for each platform
                    os.system("mkdir -p " + path)
                    # copies it to its rundir
                    os.system("make copy")
                    # creates rundir makefile with run rule:
                    #./$(PROCESSOR).$(NUMPROCESSORS).x \
                    #--load=$(SOFTWARE).$(PROCESSOR).x $(NUMPROCESSORS)
                    run_make(path, i, j, k)
    except:
        print "Error! Exiting..."
```

```
        sys.exit(1)

def run(tracker, nobuild):
    try:
        # walks the platform tracker make tree
        for i in tracker[0]:
            for j in tracker[1]:
                for k in tracker[2]:
                    rundir_path = "rundir/" + i + "." + j + "." + k + "/"
                    # tests if rundir exists
                    if not os.path.isdir(rundir_path):
                        # prints warning
                        print "WARNING! " + rundir_path + " doesn't exists!"
                        if not nobuild:
                            # creates rundir
                            build([[i],[j],[k]])
                            os.chdir(rundir_path)
                            os.system("make run")
                            os.chdir("../../")
                    else:
                        os.chdir(rundir_path)
                        os.system("make run")
                        os.chdir("../../")
    except:
        print "Error! Exiting..."
        sys.exit(1)



# === DEFINES ===
# “Smart data structures and dumb code works a lot better than the other
# way around.”
# —Eric Raymond, The Cathedral and the Bazaar, chapter 5

procs = ['mips', 'powerpc', 'sparc']
nprocs = ['01', '02', '04', '08', '16']
sws = ['basicmath', 'dijkstra','fft', 'lu', 'ocean', 'sha',
       'stringsearch', 'susancorners', 'susanedges', 'susansmoothing',
       'water', 'water-spatial']
cmds = {'-p':'all', '-n':'all', '-s':'all'}



# === MAGIC ===

# gets the command line args
cmdline=sys.argv[1:]
a = []
prep = False
nobuild = False

# armored arg passing
if len(cmdline) == 1 and cmdline[0] == "--help":
    help()
    sys.exit(0)
elif len(cmdline) == 1 and cmdline[0] == "--clean":
    clean()
    sys.exit(0)
elif len(cmdline) == 4:
    if cmdline[0] != "--build" and cmdline[0] != "--run":
        help()
        sys.exit(1)
    prep = False if cmdline[0] == "--run" else True
    for i in cmdline[1:]:
        a = i.split('=')
        if a[0] in cmds.keys():
            cmds[a[0]] = a[1]
        else:
            help()
            sys.exit(1)
elif len(cmdline) == 5:
    if cmdline[0] != "--run" or cmdline[4] != "--nobuild":
```

```
        help()
        sys.exit(1)
    nobuild = True
    for i in cmdline[1:4]:
        a = i.split('=')
        if a[0] in cmds.keys():
            cmds[a[0]] = a[1]
        else:
            help()
            sys.exit(1)
else:
    help()
    sys.exit(1)

proc = cmds['-p']
nproc = cmds['-n']
sw = cmds['-s']

# test for invalid args
if (proc not in procs and proc != "all") or \
   (nproc not in nprocs and nproc != "all") or (sw not in sws and sw != "all"):
    print "Error: invalid argument. Exiting..."
    sys.exit(1)

# platform make tracker tree creation
tracker = []
if proc == "all":
    tracker.append(procs)
else:
    tracker.append([proc])
if nproc == "all":
    tracker.append(nprocs)
else:
    tracker.append([nproc])
if sw == "all":
    tracker.append(sws)
else:
    tracker.append([sw])

rundir_path = ""

if prep:
    build(tracker)
else:
    run(tracker, nobuild)
```