Documentație proiect ASO

Faza 2

Student Ban Bogdan

Grupa 30463

Contents

Descriere soluție

S-au implementat cerințele minimale pentru verificare surselor de cod (.c) trimise de concurenți. Scriptul rulează în buclă infinită și verifică periodic (odată la 5 secunde) dacă au apărut fișiere noi pe sistem (doar fișierele cu extensia .c sunt luate în considerare). După ce soluția este primită, se crează un director care va conține în denumire data la care a fost trimisă sursa precum și numele acesteia. Directorul va conține codul sursă trimis de concurent.

Se va încerca compilarea codului cu ajutorul unui apel de sistem. In cazul în care compilarea se face cu succes, se merge mai departe. În caz contrar, se afișează un mesaj de eroare.

Se copiază pe rând 6 fișiere text care conțin date de intrare și de ieșire pentru testarea soluției. Se face verificarea pentru fiecare cuplu de fișiere, iar în cazul în care programul nu dă rezultatul așteptat, se afișează un mesaj de eroare. După realizarea celor 3 teste, se va crea un fișier text care va conține în denumire data la care o fost trimisă sursa precum și numele aceste cu prefixul "result-". Fișierul va conține textul "Correct" în caz de succes și "Wrong answer" în caz de eșec.

Exemplificare soluție

```
while True:
    for fname in os.listdir(dir_path):
        fPath = os.path.join(fname)
        if fPath.endswith('.c') and not(fPath in controlList):
            controlList.append(fPath)
            aux_path = directories_path + date.today().strftime("%Y-%m-%d") +
            time.strftime("-%H-%M-%S-", time.localtime()) + fPath[:-2]
            print(aux_path)
            try:
                os.mkdir(aux_path, 0o755)
                copyfile(dir_path + '/' + fPath, aux_path + '/' + fPath)
                test_code(aux_path,fPath)
            except OSError:
                print("Error")
    time.sleep(5)
```

În bucla infinită a scriptului se verifică periodic daca s-a încarcat un nou fișier cu extensia .c. Se folosește o listă (fPath) pentru a nu procesa decât o dată fiecare fișier. Se reține data curentă și se crează directorul concurentului. În director se copiază codul sursă. Funcția *test_code()* face verificarea codului.

```
def test_code(dirPath,ProgramName):
    # print("gcc -o " + dirPath + '/' + ProgramName[:-2] + " " + dirPath + '/' + ProgramName)
    if subprocess.call(["gcc -o " + dirPath + '/' + ProgramName[:-2] + " " + dirPath + '/' +
        ProgramName], shell = True) == 0:
        print("Successful compilation!")
        #copy input files
        copyfile("input1.txt","aso_dirs/input.txt")
        #run program
        os.system(dirPath + '/' + ProgramName[:-2])
        #compute running time
        ok=1
        #check output
        if filecmp.cmp("output1.txt","aso_dirs/output.txt") == False:
            ok=0
            print("test1 failed")
        copyfile("input2.txt","aso_dirs/input.txt")
        #run program
        os.system(dirPath + '/' + ProgramName[:-2])
        #check output
        if filecmp.cmp("output2.txt","aso_dirs/output.txt") == False:
            ok=0
            print("test2 failed")
        copyfile("input3.txt","aso_dirs/input.txt")
        #run program
        os.system(dirPath + '/' + ProgramName[:-2])
        #check output
        if filecmp.cmp("output3.txt","aso_dirs/output.txt") == False:
            ok=0
            print("test3 failed")
        aux_string = dirPath[10:29]
        #print("result-" + ProgramName[:-2] + '-' + aux_string)
        with open(dirPath + '/' + "result-" + ProgramName[:-2] + '-' + aux_string + ".txt", 'w') as
            if ok == 0:
                fp.write("Wrong answer")
            elif ok == 1:
                fp.write("Correct")
            # elif ok == 2:
```

Se compilează codul sursă folosind biblioteca *subprocess*. În caz de succes, se copiază pe rand cele 3 perechi de fișiere text pentru datele de intrare/ieșire. Se verifcă daca rezultatele sunt cele așteptate. Rezultatul verificări va fi scris într-un fișier text care va fi salvat în directorul concurentului pentru care s-a facut verificarea.

Concluzie

Faza a doua a proiectului reprezintă implementarea logicii de verificare a surselor trimise de concurenți. Rămâne importantă testarea codului pe cât mai multe cazuri pentru ca sistemul să nu poată fi exploatat. Programele cu un timp de execuție mai mare decât limita admisă (1 secundă) trebuie oprite pentru ca sistemul să nu devină supraîncărcat.