Regionális forduló 2024. november 22.

A klaszterek lényegében összekapcsolt számítógépeket jelentenek. Ha meg tudjuk oldani a futtatandó folyamatok optimális szétosztását a klaszterhez tartozó számítógépeken, akkor kívülről a klaszter egyetlen entitásnak látszik, amely rendelkezik az összes számítógép összesített erőforrásával (processzor és memória). A feladat egy olyan alkalmazás készítése, mely menedzseli a klaszteren futó folyamatokat.

A klaszter reprezentálása

A feladat során a számítógépeket mappák, a futó folyamatokat pedig fájlok fogják szimbolizálni. Például a `klaszter1` mappa tartalma a következő:

```
.
├── .klaszter
├── szamitogep2
│   ├── .szamitogep_konfig
│   ├── chrome-asdefg
│   └── word-jklbnm
└── szamitogep1
    ├── .szamitogep_konfig
    └── chrome-qwerty
```

Ebben az esetben a klaszterünkhöz jelenleg 2 darab számítógép tartozik: `szamitogep1` és `szamitogep2`. Az elsőn fut egy `chrome` és egy `word` programpéldány (vagyis folyamat), a másodikon pedig csupán egy `chrome`. A `.szamitogep_konfig` fájloktól eltekintve tehát a fájlok létezése azt jelenti, hogy egy folyamat (vagyis egy konkrét programból egy példány) éppen fut az adott számítógépen. Egy mappát pedig akkor tartunk a klaszterhez tartozó számítógépnek, amennyiben létezik benne a `.szamitogep_konfig` fájl.

A számítógépek

Természetesen a verseny során nem tudunk valódi számítógépklaszterekkel dolgozni. A számítógépeket a fenti példának megfelelően egy-egy mappa szimbolizálja, a számítógép erőforrásainak leírását pedig a mappákban lévő `.szamitogep_konfig` fájlok tartalmazzák. Ezek az erőforrások most csupán a számítógépek memória- és processzorkapacitásait jelentik. Vegyük például az alábbi `.szamitogep_konfig` fájlt:

```
2800
12000
```

Az első sor a processzor erőforrás kapacitását jelenti, mely egy egész szám. Ez az egész szám valójában a processzormagok darabszámának ezred részét jelenti, vagyis millimag. A 2800 például azt jelenti, hogy a `szamitogep2` számítógépen 2.8 processzormag elérhető. A fájl második sora szintén egy egész fájl, ami az elérhető memóriakapacitást mutatja MB-ban mérve. A példánkban 12000MB memória használható a `szamitogep2` számítógépen.

A klaszteren futó folyamatok

A futó folyamat egy konkrét programból futó példányt jelent. A klaszteren futtatandó folyamatokat fájlként reprezentáljuk, melyek a fent említett mappákban helyezkednek el például az alábbi módon:

```
.
├── chrome-asdefg
└── word-jklbnm
```

A fájlok és elhelyezkedésük tehát azt szimbolizálják, hogy az adott folyamatok mely számítógépen futnak. Fontos kiemelni, hogy egy program több példányban is futhat a klaszteren akár 1 számítógépen (mappában) is. Egy folyamat neve `<programnev>-<random 6 karakter>` melynek egyedinek kell lennie az egész klaszteren. Az ehhez tartozó fájl tartalmazza a folyamat adatait. Például:

```
2024-10-27 07:15:45
AKTÍV
100
100
```

Az első sorban az adott folyamat indításának pontos ideje található yyyy.mm.dd hh:mm:ss formátumban. Ezt követi a folyamat állapota, mely az „AKTÍV” és „INAKTÍV” szavak valamelyike aszerint, hogy a folyamat éppen rendeltetésszerűén működik-e vagy sem. A 3. és 4. sorban a folyamat által aktuálisan használt processzor és memória erőforrás található (a fent megadott mértékegységekben mérve).

Az alkalmazás

Az alkalmazásnak alkalmasnak kell lennie a klaszter menedzselésére, vagyis a meglévő folyamatok figyelésére és módosítására, valamint új programok indítására. Az, hogy a klaszteren pontosan milyen folyamatoknak kellene futniuk, a gyökér könyvtárban található `.klaszter` fájlban (adatbázisfájl) írjuk le az alábbi módon:

```
chrome
2
100
200
word
1
150
500
```

A példa fájlban az látható, hogy a klaszteren futnia kellene a „chrome” programnak 2 „AKTÍV” példányban egyenként 100 millimag processzorral és 200 MB memóriával, valamint egy „word” programnak 1 „AKTÍV” példányban 150 millimag processzorral és 100MB memóriával. A fájl sorai tehát rendre egy program neve, az adott programból futtatandó példányok darabszáma, a kért processzor- valamint a kért memóriakapacitás minden egyes programra nézve. A felhasználó számára a legfontosabb cél tehát nyilvánvalóan egy olyan állapot kialakítása, hogy minden programból fusson a kívánt példányszám valamelyik számítógépen.

Sajnos nem bízhatunk a klaszterhez tartozó számítógépek állandóságában. A gépparkunk szűkülhet és bővülhet, hiszen a számítógépek néha elromlanak, néha újak jönnek. Az alkalmazás bemenete tehát a fent említett klaszter mappa, mely tartalmazza a klaszter kívánt állapotának leírását (`.klaszter`) valamint a számítógépek mappáit. Ha új számítógépet akarunk a klaszterbe kötni, akkor ebben a gyökér könyvtárban kell egyet létrehoznunk a számítógépet jelentő mappával és benne a fent említett `.szamitogep_konfig` fájllal.

Az alkalmazás működése

Az alkalmazás induláskor kérje be a *gyökérkönyvtár* pontos elérési útját. A futás során a továbbiakban ebben kell dolgoznia. Az alkalmazásnak úgy kell működnie, hogy minden tevékenység után végrehajtja a tényleges fájlrendszer műveletet. Minden feladat után tehát újra be kell olvasni a klaszter állapotát és amennyiben szükséges, módosítani is kell. Nem elegendő a memóriában dolgozni.

Minden induláskor ellenőrizni kell, hogy a klaszter állapota megfelelő-e, vagyis, hogy az adatbázisfájlban megadott programok a kívánt példányszámban rendeltetésszerűen futnak valahol, illetve, hogy egyik számítógépen sem haladjuk meg az erőforráskapacitásokat. Egy programpéldányt akkor tekintünk rendeltetésszerűen futónak, ha létezik fájl (melynek neve a program nevével kezdődik), és az abban jelzett állapot az, hogy „AKTÍV”. Amennyiben a klaszter állapota nem megfelelő, ki kell írni, hogy pontosan melyik elv sérül:

1. Ha egy programból túl kevés „AKTÍV” állapotú példány fut, akkor ki kell írni mennyi a kívánt és aktuális „AKTÍV” és „INAKTÍV” példányszám.
2. Ha egy programból túl sok példányszám fut (a példányok státuszától függetlenül), akkor ki kell írni mennyi a kívánt és aktuális „AKTÍV” és „INAKTÍV” példányszám.
3. Amelyik számítógépeken a folyamatok összes szükséglete túllépi a számítógép valamelyik (vagy mindkettő) erőforráskapacitását, ki kell írni a számítógép nevét, valamint hogy melyik erőforrásfelhasználás sérül (processzor és/vagy memória), mennyi abból az aktuális összes felhasználás és az erőforráskapacitás. Ha mindkettő sérül, akkor mindkettőt.

Monitoring

Szükség van egy menüpontra, mellyel a felhasználó lekérheti a klaszter aktuális állapotát. Ebben meg kell jeleníteni:

1. A számítógépeket és azok kihasználtságát, vagyis a számítógépek nevét és a hozzájuk tartozó maximális és még szabad erőforrásokat.
2. A kívánt programok futó példányainak neveit, státuszait, azonosítóit és darabszámát.
3. Hány „AKTÍVT” és „INAKTÍV” folyamat fut összességében a klaszteren
4. Azt, hogy a klaszter állapota jelenleg helyes-e (mint az alkalmazás indulásánál).
5. A felhasználónak továbbá lehetőséget kell biztosítani arra, hogy egy konkrét program nevének megadása esetén az alkalmazás kiadja az összes futó példányát a következő információkkal: melyik számítógépen fut, milyen erőforrásigénnyel.

Számítógép törlése

Az alkalmazásnak lehetőséget kell biztosítania meglévő számítógépek törlésére. Ezt csak akkor szabad engedélyeznie, ha a törlendő számítógépen nem fut egyetlen folyamat sem. A felhasználónak tehát először át kell raknia a folyamatokat egy másik számítógépre. Amennyiben olyat próbál törölni, amin van futó folyamat, ki kell írni a folyamatok listáját, indításuk idejét és a státuszukat azoknak, amik még a törlendő számítógépen vannak.

Számítógép hozzáadása

Szükség van arra, hogy a klaszterhez új számítógépet lehessen hozzáadni. Ehhez be kell kérni a számítógép nevét (csak az angol ABC kis- és nagybetűit és számjegyeket tartalmazhat), valamint a számítógép erőforráskapacitását. Ezután létre kell hozni a mappát és benne a `.szamitogep_konfig` fájlt.

Egy program leállítása

Egy program leállítása azt jelenti, hogy kitöröljük a programhoz tartozó sorokat a `.klaszter` fájlból és leállítjuk az ehhez a programhoz tartozó összes futó folyamatot az összes számítógépről, vagyis töröljük a folyamatoknak megfelelő fájlokat. Természetesen csak olyan program törölhető, ami már létezik az adatbázisfájlban.

Egy program adatainak módosítása

Egy program esetében a futtatandó példányok száma, valamint a példányokhoz kért erőforrásigény módosítható. Ez a `.klaszter` fájlban a megfelelő sorok módosítását jelentik.

Egy új programpéldány futtatása

Lehetőséget kell biztosítani a felhasználónak, hogy tudjon futtatni egy új példányt egy programból. A felhasználó csak olyan programot futtathat, amire teljesülnek az alábbi kritériumok:

1. Az adatbázisfájlban (`.klaszter`) szerepel.
2. Csak olyan gépen, amit a klaszter részének tekintünk.
3. Csak abban az esetben, ha az új példánnyal nem lépjük túl a célszámítógép erőforráskapacitását (egy új példány aktuálisan használt erőforrása a `.klaszter` fájlban megadott erőforrásigény).

Az új folyamat azonosítójának egyedinek kell lennie a teljes klaszterre nézve. Az azonosító a program neve, utána egy kötőjel, valamint véletlenszerűen kiválasztott 6 alfanumerikus karakter, mely a folyamatnak megfelelő fájl neve lesz. Például: chrome-asdfgh.

Egy adott programpéldány leállítása

A leállítás kérés esetén listázni kell az éppen futó folyamatokat, melyekből a felhasználó választhat az egyedi azonosító megadásával. A folyamat leállítása a fájl törlését jelenti.

Technológiai segítség

A feladat teljes megoldásához szükségesek fájlrendszerműveletek (mappa létrehozás, mappa törlés, mappa elemeinek listázása). Hogy ezek ismeretlensége ne okozzon hátrányt, **a dokumentum végén található egy táblázat**, melyben igyekeztük összegyűjteni, hogyan lehet ezeket a műveletek megtenni az alap könyvtárak felhasználásával a különböző programozási nyelvekben.

Beadandó

Beadandó a program **forráskódja** (a programozási környezettől függően a forráskód több fájl is lehet, esetleg mappák is tartozhatnak hozzá).

A program által használt külső fájlokat mindig az induláskor bekért könyvtárba kell elhelyezni (vagy annak valamely almappájában) (A program ne tartalmazzon abszolút elérési útvonalat!)

A program mappájába készítsetek egy readme.txt állományt, amiben röviden ismertetitek a fordítás és az elindítás pontos lépéseit.

A pontozás alapfeltétele, hogy a versenybizottság egyszerűen el tudja indítani a programot.

Értékelési szempontok

A versenybizottság elsősorban a **felhasználói felületen** keresztül kipróbálva fogja pontozni a működő feature-öket, de az értékelés során a forráskód is ellenőrzésre kerül. A programkód minősége nem kerül értékelésre.

A felhasználói felületen bevitt inputokat ellenőrizni szükséges olyan mértékben, hogy helytelen input esetén a program ne álljon le programhibával vagy fusson tovább anélkül, hogy hibát jelezne.

Feltételezhetitek, hogy az input fájlok formátuma helyes.

Kevesebb, de működő funkcióval több pont érhető el, mint több funkcióval, melyek nem működnek megfelelően, kellően stabilan.

Törekedjetek arra, hogy a program minél egyszerűbben tesztelhető legyen! Ezt úgy tudjá-tok elősegíteni, ha minél egyértelműbb és könnyebb a program használata.

A döntőbe kerülő csapatok a döntőbe jutáskor e feladat folytatásaként újabb feladatokat fognak kapni, amit a döntőig kell majd megvalósítaniuk.

Jó munkát kíván a versenybizottság!

Fájlrendszer kezelés cheat sheat

	Java	C#	C++	Python
Fájl törlése	`File myFile = new File(filePath); myFile.delete();`	`File.Delete(filePath);`	`std::removefilePath);`	`os.remove (filePath)`
Mappa létrehozása	`File newDir = new File(dirPath); newDir.mkdir();`	`Directory.CreateDirectory (dirPath);`	`#include <filesystem> namespace fs = std::filesystem; fs::create_directories(dirPath)`	`os.makedirs(dirPath)`
Mappa elemeinek listázása	`File directory = new File(dirPath); File[] files = directory.listFiles();`	`string[] files = Directory.GetFiles (dirPath);`	`for (const auto & entry : fs::directory_iterator(dirPath))`	`os.listdir(dirPath)`
Üres mappa törlése	`File directory = new File(dirPath); directory.delete();`	`Directory.Delete (dirPath);`	`std::filesystem::remove(dirPath);`	`os.rmdir (dirPath)`