Eötvös Loránd Tudományegyetem

Informatikai Kar

Programozási Nyelvek és Fordítóprogramok Tanszék

Osztott rendszerek specifikációja és implementációja

IP-08bORSIG

Dokumentáció az *2.* beadandóhoz

`Szalóki Sándor`
`H8L59S`

2017. december 1.

1. Kitűzött feladat

Feladatunk annak eldöntése, hogy egy adott halmaznak létezik e olyan részhalmaza, melyben található elemek összege pontosan megegyezik egy előre megadott számmal!

A szükséges adatokat a program 3 parancssori paraméteren keresztül kapja.

Az első, egy egész értékű adat, mely a feladatban definiált számot jelzi, ezt kell valamilyen módon elérni a halmazelemek összegével.

A második, egy fájl neve - ez tartalmazza a kiinduló halmaz elemeit (inputfájl). A felépítése az alábbi: A fájl első sorában egy nemnegatív egész szám (N) áll - a kiinduló halmazunk elemszáma tehát N. A következő sorban összesen N db egész számot olvashatunk (pozitív és negatív egyaránt), melyek a halmaz elemeit jelölik (sorrendiséget nem kötünk meg köztük).

Egy megfelelő bemeneti fájl (például data.txt) ekkor:

6

3 34 4 17 5 2

A harmadik paraméter, annak a fájlnak a neve, melybe a feladat megoldása során kapott választ kell írni.

2. Felhasználói dokumentáció

A program futtatása a master futtatható fájl megnyitásával történik. A program követel paramétereket, így egyszerűbb console paranccsal indítani. Az első paraméter az elérni kívánt összeg, a második a bemeneti számokat tartalmazó fájl neve, illetve a harmadik a kimeneti fájl neve.

2.1. Rendszer-követelmények, telepítés

A programunk több platformon is futtatható, dinamikus függősége nincsen, bármelyik, manapság használt PC-n működik. Külön telepíteni nem szükséges, elég a futtatható állományt elhelyezni a számítógépen.

2.2. A program használata

A program használata egyszerű, de célszerű paranccsorból indítani, hiszen szükséges a futtatáshoz a paraméterek megadása. Két futtatható állomány (master és child) közül a master állományt kell futtatni, de mindkettő fájl jelenléte szükséges. Az első paraméter az elérni kívánt összeg, a második a bemeneti számokat tartalmazó fájl neve, illetve a harmadik a kimeneti fájl neve.

3. Fejlesztői dokumentáció

A program két programból áll. Egy master főprogramból, illetve egy child gyerekprogramból. PVM segítségével a master a megfelelő adatok megszerzése után kiadja a megoldás feladatát a child programnak. Ekkor a child az alapadattól függően eredményt küld vissza a master programnak, vagy további child programokat indít magából amíg az egyik ilyen child megoldást nem talál (ha létezik megoldás).

3.1. Megoldási mód

A program két részre lett osztva. A master program először beolvassa a fájlból a szükséges adatokat, majd az összes adatot átadja az 1. child programnak. A child program ekkor megnézi, hogy a felada készen van-e, amennyiven nincs, további 2 child programot indít magából, de azokat már önmagától kisebb feladatokkal látja el. Minden child program további programokat hoz létre egészen addig, ameddig a child program már az előtt az alapfeladat előtt áll, amiről egyértelműen eltudja dönteni, hogy megoldás-e, vagy sem. Miután ezeknek a számolásoknak vége, a bináris fát alkotott child programokon felfelé haladva megnézzük, hogy melyik ágon van megoldás. Amennyiben bármelyik ágon található megoldás, az 1. child program ezt visszaküldi a master programnak, amely a végén ezt kiírja a megfelelő fájlba.

3.2. Implementáció

A megoldási módban leírtakat 2 fájlba szervezve oldjuk meg. master.cpp felelős a master progarmért, child.cpp felelős a child programért. A két program közötti kommunikációt PVM biztosítja. A gyorsaság és biztonság érdekében a legtöbb változót a stack-en tárolunk, illetve C tömbök helyett std::vector<T>-t használunk. Mivel a PVM függvényei C típusokat várnak, az std::vector<T> adatai küldése előtt ebből egy lokális C tömböt hozunk létre. A két program közötti kommunikáció egyszerű és minimális. A master program elküldi az 1. child programnak az összeg értékét, a tömb hosszát, illetve a tömb értékeit. A child process ebből el tudja dönteni, hogy a feladat már megoldott-e, ha nem akkor további 2 child programot indít, aminek elküldi ugyan ezeket az adatokat, de már egy kisebb tömbre. Minden child process elküldi a szülő programjának, hogy talált-e megoldást. Ha a szülő program egy child, akkor 2 ilyen eredményt kap, ezek közül ha bármelyik megoldás, akkor az ő továbbküldött értéke is ezt fogja mutatni. Ahogy visszatérünk az 1. child programhoz, az már tudja, hogy található-e megoldás. Ezt az értéket elküldi a master programnak.

3.3. Fordítás menete

A fordításhoz szükséges egy Makefail.aimk, amely tartalmazza a szükséges paramétereket, illetve beállításokat. Ezzel fordítanunk kell a master.cpp, illetve child.cpp fájlt is.

3.4. Tesztelés

A program teszteléséhez több configuráción lett futtatva. Ehhez PVM-ben a blade-ek számát módosítottuk.

Blade-ek száma	Másodperc
1	0.0697199
2	0.0445748
3	0.0523889
4	0.0549406
5	0.0359066
6	0.0407191