Álmos Totálisan Jóarc Ütemezője

1.0

Készítette Doxygen 1.9.1

1. fejezet

Névtérmutató

1.1. Névtérlista

Az összes dokumentált névtér listája rövid leírásokkal:

2. fejezet

Osztálymutató

2.1. Osztálylista

Az összes osztály, struktúra, unió és interfész listája rövid leírásokkal:

3. fejezet

Névterek dokumentációja

3.1. almtree névtér-referencia

A piros-fekete fa színeit tartalmazó névtér.

Enumerációk

- enum **COLOR** { **RED** , **BLACK** }

3.1.1. Részletes leírás

A piros-fekete fa színeit tartalmazó névtér.

4. fejezet

Osztályok dokumentációja

4.1. Drawer osztályreferencia

Az információk megjelenítésért felelős osztály.

`#include <drawer.hpp>`

Publikus tagfüggvények

- **Drawer** (RbTree *tree, std::mutex *mutex)
- void init ()

  Inicializálja a rajzolási felületet.
- void drawRunnable ()

  A rajzolást vezérlő runnable függvény.
- void disableDraw ()

  Megállítja a rajzolási folyamatot.
- void callReDraw ()

  Újrarajzolja a piros-fekete fa állapotát.
- void setRunningTask (std::shared_ptr< Task > currentTask)

  Beállítja az éppen futó folyamatot.

Privát tagfüggvények

- void draw ()

  A rajzolómetódus.
- int treeCToGraphicsC (almtree::COLOR color)

  Átkonvertálja az almtree névtérben levő színt a graphics.h színére.

Privát attribútumok

- RbTree * **tree**
- std::mutex * **mutex**
- std::shared_ptr< Task > currentTask

  Az éppen háttérben futó folyamat.
- int **gd**
- int **gm**
- int **window**
- bool **shouldDraw**
- bool **reDraw**

4.1.1. Részletes leírás

Az információk megjelenítésért felelős osztály.

4.1.2. Tagfüggvények dokumentációja

4.1.2.1. draw()

```
void Drawer::draw ( )  [private]
```

A rajzolómetódus.

Lerajzolja a piros-fekete aktuális állapotát jelezve a csomópontokban az eltelt időt amennyit futott a jelzett folyamat, és a folyamat PID-jét. A képernyő alján ugyanakkor kiírja, hogy éppen melyik folyamat fut.

4.1.2.2. drawRunnable()

```
void Drawer::drawRunnable ( )
```

A rajzolást vezérlő runnable függvény.

Addig fut, amíg a shouldDraw változó értéke true. Akkor rajzoltat, ha a reDraw értéke true.

4.1.2.3. treeCToGraphicsC()

```
int Drawer::treeCToGraphicsC (
            almtree::COLOR color )  [private]
```

Átkonvertálja az almtree névtérben levő színt a graphics.h színére.

Paraméterek

color	Egy szín az almtree névtérből (vagy piros vagy fekete).

Visszatérési érték

A graphics.h COLOR enumjának egy tagja (vagy piros vagy fekete).

Ez a dokumentáció az osztályról a következő fájl alapján készült:

- headers/drawer.hpp

4.2. Message struktúrareferencia

A szerverhez intézett üzenet.

```
#include <message.h>
```

Publikus attribútumok

- MSG_TYPE **type**
- int **pTime**
- char **programName** [50]

4.2.1. Részletes leírás

A szerverhez intézett üzenet.

Ez a dokumentáció a struktúráról a következő fájl alapján készült:

- headers/message.h

4.3. Node struktúrareferencia

A csomópont a piros fekete fában.

`#include <node.hpp>`

Publikus tagfüggvények

- **Node** (int key)
- **Node** (const Node &cpy)
- Node * sibling ()

  Visszatéríti a csomópont testvérét.
- bool isOnLeft ()

  Ellenőrzi, hogy a csomópont a szülője bal oldalán van-e.

Publikus attribútumok

- int **key**
- std::shared_ptr< void > **data**
- almtree::COLOR **color**
- Node * **left**
- Node * **right**
- Node * **parent**

4.3.1. Részletes leírás

A csomópont a piros fekete fában.

4.3.2. Tagfüggvények dokumentációja

4.3.2.1. isOnLeft()

```
bool Node::isOnLeft ( )
```

Ellenőrzi, hogy a csomópont a szülője bal oldalán van-e.

Visszatérési érték

true ha a bal oldalon van, másképpp false

4.3.2.2. sibling()

```
Node* Node::sibling ( )
```

Visszatéríti a csomópont testvérét.

Visszatérési érték

A csomópont testvére.

Ez a dokumentáció a struktúráról a következő fájl alapján készült:

- headers/node.hpp

4.4. RbTree osztályreferencia

Nagyjából standard és általános piros-fekete fa implementáció.

```
#include <rbtree.hpp>
```

Publikus tagfüggvények

- Node * rbInsert (const int &key)
  Behelyez egy új csomópontot a piros-fekete fába, és frissíti a minimális csomópontot.
- void rbDelete (Node *node)
  Kivesz egy csomópontot a piros-fekete fából.
- std::vector< Node * > getLevelOrder ()
  Visszatéríti a fa csomópontjait szint szerinti sorrendben.
- Node * getRoot ()
  Visszatéríti a piros-fekete fa gyökerét.
- Node * search (const int &key)
  Visszatéríti a fa "key" kulcsú csomópontját, ha létezik.
- std::shared_ptr< void > popMin ()
  Kitörli a minimális csomópontot, és visszatéríti a benne levő adatot.

Privát tagfüggvények

- Node ∗ insert (Node ∗root, Node ∗newNode)

  Privát rekurzív bináris keresőfa beszúrás.
- void fixTreeAfterInsert (Node ∗&root, Node ∗newNode)

  Kijavítja beszúrás után a piros-fekete tulajdonságot.
- void rotateLeft (Node ∗&root, Node ∗&nodePtr)

  Végrehajt egy bal forgatást a megadott csomóponton.
- void rotateRight (Node ∗&root, Node ∗&nodePtr)

  Végrehajt egy jobb forgatást a megadott csomóponton.
- Node ∗ replacementSearch (Node ∗x)

  Megkeresi az x csomópont utódját BST törlés esetén.
- Node ∗ successor (Node ∗x)

  Visszatéríti az x csomópont által jelzett al-fa legkisebb elemét.
- void fixDoubleBlack (Node ∗x)

  Kijavítja a megadott x csúcs "duplafeketeségét".

Privát attribútumok

- Node ∗ **root**
- Node ∗ **minNode**

4.4.1. Részletes leírás

Nagyjából standard és általános piros-fekete fa implementáció.

4.4.2. Tagfüggvények dokumentációja

4.4.2.1. fixDoubleBlack()

```
void RbTree::fixDoubleBlack (
            Node * x )  [private]
```

Kijavítja a megadott x csúcs "duplafeketeségét".

A dupla-fekete javításnál főleg a testvér színét figyeljük. a) Ha a testvér fekete, és van legalább egy piros gyermeke, akkor maximum két forgatással ki lehet javítani a fekete magasságot. b) Ha a testvér fekete és mindkét gyermeke fekete, akkor átszínezzük a testvért pirosra, a szülőt feketére, s ha a szülő duplafekete lesz, meghívjuk rá is a függvényt. c) Ha a testvér piros, akkor átszínezzük feketére, a szülőt pirosra, forgatunk egyet a szülőn attól függően, hogy hol van a testvér, és visszatérünk az a) vagy b) esetre.

Paraméterek

x	A csúcs amit "megjavítunk".

4.4.2.2. fixTreeAfterInsert()

```
void RbTree::fixTreeAfterInsert (
            Node *& root,
            Node * newNode )  [private]
```

Kijavítja beszúrás után a piros-fekete tulajdonságot.

A kijavítási folyamat tartalmazhat forgatásokat és átszínezéseket, és rotációkat. Mivel minden beszúrt új csomópont piros, ezért leredukálható a probléma az egymást követő piros eset kijavítására. Ha a nagybácsi (a newNode szülőjének testvére) piros, akkor egyszerűen a szülő s a nagybácsi színét feketére állítjuk és a nagyszülőjét pedig pirosra. Fekete esetben 4 eset fordulhat elő:

1. Bal-bal(a szülő a nagyszülő bal gyermeke és a newNode a szülő bal gyermeke): Végrehajtunk egy jobb forgatást a nagyszülőn, kicseréljük a szülő és nagyszülő színét.
2. Bal-jobb: Végrehajtunk egy bal forgatást a szülőn, ezután visszatértünk a bal-bal esethez.
3. Jobb-jobb: Végrehajtunk egy bal forgatást a nagyszülőn, kicseréljük a szülő és a nagyszülő színét.
4. Jobb-bal: Végrehajtunk egy jobb forgatást a szülőn, ezután visszatértünk a jobb-jobb esethez. A fenti eseteket addig folytatjuk, amíg a newNode paraméter fekete nem lesz, vagy pedig a szülő fekete lesz.

Paraméterek

root	A fa gyökere
newNode	Az új beszúrt csomópont.

4.4.2.3. getLevelOrder()

```
std::vector<Node*> RbTree::getLevelOrder ( )
```

Visszatéríti a fa csomópontjait szint szerinti sorrendben.

A getLevelOrder függvény egy várakozási sor segítségével bejárja a piros-fekete fát, szintről-szintre haladva.

Visszatérési érték

A fa csomópontjait tartalmazó vektor.

4.4.2.4. getRoot()

```
Node* RbTree::getRoot ( )
```

Visszatéríti a piros-fekete fa gyökerét.

Visszatérési érték

A fa gyökerét tartalmazó csomópont.

4.4.2.5. insert()

```
Node* RbTree::insert (
            Node * root,
            Node * newNode )  [private]
```

Privát rekurzív bináris keresőfa beszúrás.

A szülőt azért térítjük vissza, hogy meg tudjuk hívni a piros-fekete tulajdonság kijavítására szolgáló fixTreeAfter↩ Insert függvényt.

Visszatérési érték

Visszatéríti az újonnan beszúrt csomópont szülőjét.

4.4.2.6. popMin()

```
std::shared_ptr<void> RbTree::popMin ( )
```

Kitörli a minimális csomópontot, és visszatéríti a benne levő adatot.

Visszatérési érték

A csomópontban levő adatra mutató megosztott mutató.

4.4.2.7. rbDelete()

```
void RbTree::rbDelete (
            Node * node )
```

Kivesz egy csomópontot a piros-fekete fából.

Először kitörli egy standard BST törléssel, utána pedig, ha fennáll az eset, kijavítja a dupla fekete hibát a fix↩ DoubleBlack függvénnyel. Hívjuk v-nek a törölt csúcsot és u-nak a csúcsot, amivel lecseréljük ezt a törölt csúcsot. A legegyszerűbb eset akkor áll fenn, ha vagy u vagy v piros, ekkor egyszerűen az u csúcsot átszínezzük pirosra. Ha mindkettő csúcs fekete, akkor törlés után (vagy előtt, ha a leváltó nullptr) meghívjuk a fixDoubleBlack függvényt.

Paraméterek

node	A törlendő csomópontra mutató pointer.

4.4.2.8. rbInsert()

```
Node* RbTree::rbInsert (
```

```
            const int & key )
```

Behelyez egy új csomópontot a piros-fekete fába, és frissíti a minimális csomópontot.

Az rbInsert függvény először létrehoz egy új csomópontot, majd végrehajt egy standard bináris-keresőfa beszúrást az insert metódussal. utána pedig (ha szükséges) kijavítja az esetleges piros-fekete tulajdonság megsértését a fixTreeAfterInsert privát metódussal.

Paraméterek

key	A kulcs, ami alapjan el lesz rendezve a csomópont a keresőfában.

Visszatérési érték

Az újonnan létrehozott csomópont.

4.4.2.9. replacementSearch()

```
Node* RbTree::replacementSearch (
            Node * x )  [private]
```

Megkeresi az x csomópont utódját BST törlés esetén.

Paraméterek

x	A csomópont aminek a "leváltóját" keressük.

Visszatérési érték

Az x csomópont "leváltója".

4.4.2.10. rotateLeft()

```
void RbTree::rotateLeft (
            Node *& root,
            Node *& nodePtr )  [private]
```

Végrehajt egy bal forgatást a megadott csomóponton.

Paraméterek

root	A fa gyökere.
nodePtr	A "megforgatott" csomópont.

4.4.2.11. rotateRight()

```
void RbTree::rotateRight (
            Node *& root,
            Node *& nodePtr )  [private]
```

Végrehajt egy jobb forgatást a megadott csomóponton.

Paraméterek

root	A fa gyökere.
nodePtr	A "megforgatott" csomópont.

4.4.2.12. search()

```
Node* RbTree::search (
            const int & key )
```

Visszatéríti a fa "key" kulcsú csomópontját, ha létezik.

Paraméterek

key	A keresett csomópont kulcsa.

Visszatérési érték

A kulcsot tartalmazó csomópont, másképp egy nullptr.

4.4.2.13. successor()

```
Node* RbTree::successor (
            Node * x )  [private]
```

Visszatéríti az x csomópont által jelzett al-fa legkisebb elemét.

Paraméterek

x	Az x csomópont.

Visszatérési érték

Az al-fa legbaloldalibb vagy legkisebb nem-nullptr eleme.

Ez a dokumentáció az osztályról a következő fájl alapján készült:

- headers/rbtree.hpp

4.5. Server osztályreferencia

A folyamatokat kezelő, vizualizációt feladatfuttatással összekötő szerver.

`#include <server.hpp>`

Publikus tagfüggvények

- **Server** (RbTree *tree, std::mutex *mutex, Drawer *drawer)
- void **listenRunnable** ()
- void **runTaskRunnable** ()

Privát tagfüggvények

- void insertTask (const std::shared_ptr< Task > &task)

  Beilleszt egy folyamatot menedzseltek közé.
- void removeTask (int pid)

  Eltávolít egy folyamatot a PID-je alapján.
- void runTask ()

  Futtatja a minimális vruntime-al rendelkező folyamatot.
- void updateOwners ()

  Frissíti a folyamatok csomópontjait törlés után.

Privát attribútumok

- int pidCount

  A már inicializált PID-eket számolja.
- std::mutex * **mutex**
- Message message

  Az éppen feldolgozandó üzenet.
- io_context **context**
- udp::endpoint **receiver**
- udp::socket **serverSocket**
- mutable_buffer **buffer**
- err_code **ec**
- RbTree * **tree**
- Drawer * **drawer**
- bool **shouldRun**
- std::unordered_map< int, std::shared_ptr< Task > > tasks

  A szerver által menedzselt feladatok.

4.5.1. Részletes leírás

A folyamatokat kezelő, vizualizációt feladatfuttatással összekötő szerver.

4.5.2. Tagfüggvények dokumentációja

4.5.2.1. insertTask()

```
void Server::insertTask (
            const std::shared_ptr< Task > & task )  [private]
```

Beilleszt egy folyamatot menedzseltek közé.

Paraméterek

task	A beillesztendő feladatot kezelő mutató

4.5.2.2. removeTask()

```
void Server::removeTask (
            int pid )  [private]
```

Eltávolít egy folyamatot a PID-je alapján.

A függvény megkeresi a tasks hasítótáblában a folyamatot, majd pedig eltávolítja a piros-fekete fából is.

Paraméterek

pid	A folyamat Process ID-ja

4.5.2.3. runTask()

```
void Server::runTask ( )  [private]
```

Futtatja a minimális vruntime-al rendelkező folyamatot.

A függvény az rbTree popMin metódusát felhasználva megkapja a minimális vruntime-al rendelkező folyamatot, majd pedig "futtatja" --> A task pTime változó ideéig sleepel a folyamatot futtató thread.

4.5.2.4. updateOwners()

```
void Server::updateOwners ( )  [private]
```

Frissíti a folyamatok csomópontjait törlés után.

Ez a függvény egy szükségszerű megoldás arra az eshetőségre, mikor a fából való törlés a másoló módszer alapján dolgozik: ilyenkor egy Task struktúra treeNode pointere olyan csomópontra mutathat, ami már esetleg törölve volt.

Ez a dokumentáció az osztályról a következő fájl alapján készült:

- headers/server.hpp

4.6. Task osztályreferencia

Egy Unix process megtestesitője.

```
#include <task.hpp>
```

Publikus tagfüggvények

- **Task** (int pid, std::string &name, int pTime, int vruntime)
- std::string **getName** ()
- Node ∗ **getNode** ()
- int **getPid** ()
- int **getVRuntime** ()
- void **setNode** (Node ∗treeNode)
- void **setXCoord** (int x_coord)
- void **setYCoord** (int y_coord)
- void **setVRunTime** (int vruntime)
- void **setLevel** (int level)
- int **getXCoord** ()
- int **getYCoord** ()
- void **initLevel** ()
- int **getLevel** ()
- int **getPTime** ()

Privát attribútumok

- int pid

  A feladat folyamat azonosítója.
- int x_coord

  A feladat koordinátái a megjelenített fában.
- int **y_coord**
- std::string **name**
- Node ∗ treeNode

  A feladat csomópontja a piros-fekete fában;.
- int level

  A feladat szintje a piros-fekete fában.
- int vruntime

  A feladat összes futási ideje.
- int pTime

  A feladat futási ideje egy iterációban.

4.6.1. Részletes leírás

Egy Unix process megtestesitője.

Ez a dokumentáció az osztályról a következő fájl alapján készült:

- headers/task.hpp

Tárgymutató