Simulació del rendiment de processadors interconectats - Entrega final

Generat per Doxygen 1.9.1

Chapter 1

Índex de Classes

1.1 Llista de Classes

Aquestes són les classes, estructures, unions i interfícies acompanyades amb breus descripcions:

Chapter 2

Índex de Fitxers

2.1 Llista dels Fitxers

Aquesta és la llista de tots els fitxers acompanyats amb breus descripcions:

Chapter 3

Documentació de les Classes

3.1 Referència de la Classe cluster

Representa un cluster en forma d'arbre binari. Està format per processadors i processos.

`#include <cluster.hh>`

Classes

- struct info

Mètodes públics

- cluster ()

  Crea un nou cluster inicialitzat.
- cluster (Procesor &p, cluster &le, cluster &ri)

  Crea un nou cluster format pel pare p i els clusters fills le i ri.
- void read_cluster ()

  Llegeix els components d'un cluster.
- void modify_processor (string id, const Procesor &p)

  Canvia les dades del processador amb identificador id per les del processador p.
- void insert_cluster (const map< string, Procesor > &newmap, const BinTree< string > &newbintree, const string &p)

  Insereix una nova branca al cluster.
- int add_proces_directly (string id, Proces &p)

  Intenta afegir el procès p al processador amb identificador id.
- void increase_time (int t)

  Incrementa el temps del rellotge t unitats de temps.
- void compact_memory ()

  Compacta la memòria de tots els processadors del cluster.
- int remove_proces (string idprocesor, string idproces)

  Elimina un procès d'un processador.
- void write_procesors ()

  Escriu els processados al canal de sortida.
- bool exist (string id)

Retorna si el processador amb identificador existeix.

- bool aux_procesors (const string &p)

  Retorna una parella amb els processadors auxiliars de p.
- Procesor search_procesor (string id)

  Cerca el processador amb identificador id.
- void write_structure ()

  Escriu l'estructura del cluster al canal de sortida.
- BinTree< string > bintree_cluster ()
- map< string, Procesor > map_cluster ()
- string where_add (string id, int mem)

Mètodes Privats

- int best2 (info a, info b)
- int best3 (info a, info b, info c)
- void write_i_procesor (const BinTree< string > &cl)
- void write_i_struct (const BinTree< string > &cl)
- void read (BinTree< string > &c)
- bool search (const BinTree< string > &cl, const string &p)
- BinTree< string > add (string k, const BinTree< string > &a, const BinTree< string > &n)
- void where_i_add (const BinTree< string > &t, info &space, int mem, string idproces)

Atributs Privats

- BinTree< string > c
- map< string, Procesor > m
- int time

3.1.1 Descripció Detallada

Representa un cluster en forma d'arbre binari. Està format per processadors i processos.

Definició a la línia 24 del fitxer cluster.hh.

3.1.2 Documentació del Constructor i el Destructor

3.1.2.1 cluster() [1/2]

```
cluster::cluster ( )
```

Crea un nou cluster inicialitzat.

Precondició

true

Postcondició

S'ha inicialitzat un nou cluster.

Definició a la línia 6 del fitxer cluster.cc.

```
6                {
7     string dada = "0";
8     c = BinTree<string>(dada);
9     time = 0;
10 }
```

3.1.2.2 cluster() [2/2]

```
cluster::cluster (
            Procesor & p,
            cluster & le,
            cluster & ri )
```

Crea un nou cluster format pel pare p i els clusters fills le i ri.

Precondició

true

Postcondició

S'ha inicialitzat un nou cluster format per dos clusters le i ri units per un pare p.

Paràmetres

p	Processador pare
le	Fill esquerre del cluster
ri	Fill dret del cluster

3.1.3 Documentació de les Funcions Membre

3.1.3.1 add()

```
BinTree< string > cluster::add (
            string k,
            const BinTree< string > & a,
            const BinTree< string > & n )  [private]
```

Definició a la línia 136 del fitxer cluster.cc.

```
136                                                                                                    {
137 /* Pre: cert */
138 /* Post: El valor de cada node del resultat és la suma del valor del node
139 corresponent d'a i el valor k */
140 if (a.empty()) return BinTree<string>();
141 if (a.value() == k) return n;
142 else {
143 return BinTree<string>(a.value(), add(k, a.left(), n), add(k, a.right(), n));
144 }
145 }
```

3.1.3.2 add_proces_directly()

```
int cluster::add_proces_directly (
              string id,
              Proces & p )
```

Intenta afegir el procès p al processador amb identificador id.

Precondició

true

Postcondició

S'afegeix el procès p al procesador id.

Paràmetres

id	Identificador del processador on es vol inserir el procès.
p	Procès que es vol inserir.

Retorna

true si el procès s'ha insertat al processador o aquest no existeix al cluster.

false si el procès no es pot insertar al processador o aquest no existeix al cluster.

Definició a la línia 30 del fitxer cluster.cc.

```
30                                                  {
31     if (m.find(idpro) == m.end()) return 1;
32     return m[idpro].add_proces(p);
33 }
```

3.1.3.3 aux_procesors()

```
bool cluster::aux_procesors (
              const string & p )
```

Retorna una parella amb els processadors auxiliars de p.

Precondició

El processador p existeix al cluster.

Postcondició

Retorna els processadors auxiliars de p. Si un d'aquests està buit retorna processadors amb identificador -1.

Paràmetres

p	Procesador que existeix al cluster

Retorna

pair<Procesor, Procesor>

Definició a la línia 66 del fitxer cluster.cc.

```
66                                          {
67     return search(c, p);
68 }
```

3.1.3.4 best2()

```
int cluster::best2 (
              info a,
              info b )  [private]
```

Definició a la línia 240 del fitxer cluster.cc.

```
240                                 {
241     if (not a.cap) return 2;
242     if (not b.cap) return 1;
243     if (not a.cap and not b.cap) return 1;
244     if (a.forat == b.forat and a.freemem == b.freemem) return 1;
245     if (a.forat < b.forat) return 1;
246     if (a.forat > b.forat) return 2;
247     if (a.freemem > b.freemem) return 1;
248     if (a.freemem < b.freemem) return 2;
249     return 0;
250  }
```

3.1.3.5 best3()

```
int cluster::best3 (
              info a,
              info b,
              info c )  [private]
```

Definició a la línia 252 del fitxer cluster.cc.

```
252                                         {
253     int n = best2(a, b);
254     if (n == 1){
255         n = best2(a, b);
256         if (n == 1) return 1;
257         else return 2;
258     }
259     n = best2(b, c);
260     if (n == 1) return 2;
261     return 3;
262
263 }
```

3.1.3.6 bintree_cluster()

```
BinTree< string > cluster::bintree_cluster ( )
```

Definició a la línia 146 del fitxer cluster.cc.

```
146                                         {
147     return c;
148 }
```

3.1.3.7 compact_memory()

```
void cluster::compact_memory ( )
```

Compacta la memòria de tots els processadors del cluster.

Precondició

El cluster està inicialitzat.

Postcondició

Es retorna el cluster amb la memòria de tots els processadors compactada.

Definició a la línia 42 del fitxer cluster.cc.

```
42                                  {
43     for (auto it = m.begin(); it != m.end(); ++it){
44         it->second.compact_memory();
45     }
46 }
```

3.1.3.8 exist()

```
bool cluster::exist (
             string id )
```

Retorna si el processador amb identificador existeix.

Precondició

Id es un string d'identificador vàlid

Postcondició

Retorna si el processador id existeix

Paràmetres

id	Identificador del processador

Retorna

true si el processador existeix.

false si el processador no existeix.

Definició a la línia 61 del fitxer cluster.cc.

```
61                                  {
62     if (m.find(id) == m.end()) return false;
63     return true;
64 }
```

3.1.3.9 increase_time()

```
void cluster::increase_time (
             int t )
```

Incrementa el temps del rellotge t unitats de temps.

Precondició

true

Postcondició

T = T + t.

Paràmetres

t	Unitats de temps que han transcorregut.

Definició a la línia 35 del fitxer cluster.cc.

```
35                                      {
36     time += t;
37     for (auto i = m.begin(); i != m.end(); i++){
38         i->second.increase_time(t);
39     }
40 }
```

3.1.3.10 insert_cluster()

```
void cluster::insert_cluster (
             const map< string, Procesor > & newmap,
             const BinTree< string > & newbintree,
             const string & p )
```

Insereix una nova branca al cluster.

Precondició

El processador arrel de c existeix a C, no te processos en execució ni processadors auxiliars.

Postcondició

Es retorna el cluster C amb el cluster c adherit al pricessador arrel.

Paràmetres

c	Cluster d'entrada amb un processador id que ja està dintre del cluster C

Definició a la línia 24 del fitxer cluster.cc.

```
24
                {
25     c = add(p, c, newbintree);
26     m.insert(newmap.begin(), newmap.end());
27     m.erase(p);
28 }
```

3.1.3.11 map_cluster()

```
map< string, Procesor > cluster::map_cluster ( )
```

Definició a la línia 150 del fitxer cluster.cc.

```
150                                          {
151     return m;
152  }
```

3.1.3.12 modify_processor()

```
void cluster::modify_processor (
             string id,
             const Procesor & p )
```

Canvia les dades del processador amb identificador id per les del processador p.

Precondició

L'identificador de p == id i existeix un procesador al cluster amb identificador id.

Postcondició

Paràmetres

id	Identificador del processador que es vol modificar
p	Processador pel qual es vol modificar

Definició a la línia 19 del fitxer cluster.cc.

```
19                                                  {
20     auto it = m.find(id);
21     it -> second = p;
22 }
```

3.1.3.13 read()

```
void cluster::read (
            BinTree< string > & c )  [private]
```

Definició a la línia 111 del fitxer cluster.cc.

```
111                                     {
112     string s;
113     int num;
114     cin » s;
115     if (s != "*"){
116         cin » num;
117         m[s]=Procesor(s, num);
118         BinTree<string> le, ri;
119         read(le);
120         read(ri);
121         cl = BinTree<string>(s, le, ri);
122     }
123
124 }
```

3.1.3.14 read_cluster()

```
void cluster::read_cluster ( )
```

Llegeix els components d'un cluster.

Precondició

El cluster està inicialitzat i/o conte dades.

Postcondició

El cluster actual conté les dades introduides a l'entrada.

Si el cluster C contenía dades, aquestes s'han eliminat.

Definició a la línia 14 del fitxer cluster.cc.

```
14                          {
15     read(c);
16
17 }
```

3.1.3.15 remove_proces()

```
int cluster::remove_proces (
            string idprocesor,
            string idproces )
```

Elimina un procès d'un processador.

Paràmetres

idprocesor	Identificador del processador.
idproces	Identificador del procès.

Retorna

true si el procès s'ha afegit al processador.

false si el procès no es pot afegir al processador.

Definició a la línia 48 del fitxer cluster.cc.

```
48                                                              {
49     if (m.find(idprocesor) == m.end()) return 1;
50     if (not m[idprocesor].remove_proces(idproces)) return 2;
51     return 0;
52 }
```

3.1.3.16 search()

```
bool cluster::search (
            const BinTree< string > & cl,
            const string & p )  [private]
```

Definició a la línia 126 del fitxer cluster.cc.

```
126                                                             {
127     if (cl.empty()) return true;
128     if (cl.value() == p){
129         if (cl.left().empty() and cl.right().empty()) return true;
130         return false;
131     }
132     return search(cl.left(), p) or search(cl.right(), p);
133
134 }
```

3.1.3.17 search_procesor()

```
Procesor cluster::search_procesor (
            string id )
```

Cerca el processador amb identificador id.

Precondició

El procesador amb identificador id existeix al cluster.

Postcondició

Es retorna el processador id.

Paràmetres

id	Identificador del processador

Retorna

Procesor

Definició a la línia 70 del fitxer cluster.cc.

```
70                                          {
71     return m[id];
72 }
```

3.1.3.18 where_add()

```
string cluster::where_add (
              string id,
              int mem )
```

Definició a la línia 79 del fitxer cluster.cc.

```
79                                           {
80     info best;
81     where_i_add(c, best, mem, id);
82     if (best.cap) return best.idprocesor;
83     return "...";
84 }
```

3.1.3.19 where_i_add()

```
void cluster::where_i_add (
              const BinTree< string > & t,
              info & space,
              int mem,
              string idproces )  [private]
```

Definició a la línia 154 del fitxer cluster.cc.

```
154                                                                                                                 {
155     if(t.empty()){      // cas base t es buit
156         space.cap = false;
157         return;
158     }
159     if(t.left().empty() and t.right().empty()){   // cas base t no te fills
160         if(not m[t.value()].exist(idproces)){
161             pair <int, int> p = m[t.value()].fit(mem);
162             if(p.first == -1){
163                 space.cap = false;
164                 return;
165             }
166             space.idprocesor = t.value();
167             space.forat = p.first;
168             space.freemem = p.second;
169             space.cap = not m[t.value()].exist(idproces);
170             return;
171         }
172         else {
173             space.cap = false;
174             return;
175         }
176     }
177     // pas inductiu
178     if (t.left().empty() xor t.right().empty()){    // cas inductiu t té 1 fill
179         info bestvalue, bestson;
180         pair <int, int> p = m[t.value()].fit(mem);
181         if (p.first == -1){
182             bestvalue.cap = false;
183         }
184         else {
185             bestvalue.idprocesor = t.value();
186             bestvalue.forat = p.first;
187             bestvalue.freemem = p.second;
188             bestvalue.cap = not m[t.value()].exist(idproces);
189         }
190         if(t.left().empty()){
191             where_i_add(t.right(), bestson, mem, idproces);
192         }
```

```
193         else where_i_add(t.left(), bestson, mem, idproces);
194         if (best2(bestvalue, bestson) == 1){
195             space = bestvalue;
196             return;
197         }
198         else{
199             space = bestson;
200             return;
201         }
202     }
203     // cas inductiu t té 2 fills
204     info bestvalue, bestle, bestri;
205     pair <int, int> p = m[t.value()].fit(mem);
206     if (p.first == -1){
207         bestvalue.cap = false;
208     }
209     else {
210         bestvalue.idprocesor = t.value();
211         bestvalue.forat = p.first;
212         bestvalue.freemem = p.second;
213         bestvalue.cap = not m[t.value()].exist(idproces);
214     }
215     where_i_add(t.left(), bestle, mem, idproces);
216     where_i_add(t.right(), bestri, mem, idproces);
217     if(best2(bestle, bestri) == 1){
218         if (best2(bestvalue, bestle) == 1){
219             space = bestvalue;
220             return;
221         }
222         else{
223             space = bestle;
224             return;
225         }
226     }
227     else {
228         if (best2(bestvalue, bestri) == 1){
229             space = bestvalue;
230             return;
231         }
232         else{
233             space = bestri;
234             return;
235         }
236     }
237     return;
238     }
```

3.1.3.20 write_i_procesor()

```
void cluster::write_i_procesor (
            const BinTree< string > & cl )  [private]
```

Definició a la línia 86 del fitxer cluster.cc.

```
86                                                  {
87     if (not cl.empty()){
88         cout « cl.value() « endl;
89         m[cl.value()].write();
90         BinTree<string> le = cl.left();
91         BinTree<string> ri = cl.right();
92         write_i_procesor(le);
93         write_i_procesor(ri);
94     }
95 }
```

3.1.3.21 write_i_struct()

```
void cluster::write_i_struct (
            const BinTree< string > & cl )  [private]
```

Definició a la línia 98 del fitxer cluster.cc.

```
98                                                  {
99     if (cl.empty()) cout « " ";
100     else{
101         cout « "(";
102         cout « cl.value();
103         BinTree<string> le = cl.left();
104         BinTree<string> ri = cl.right();
105         write_i_struct(le);
106         write_i_struct(ri);
107         cout « ")";
108     }
109 }
```

3.1.3.22 write_procesors()

```
void cluster::write_procesors ( )
```

Escriu els processados al canal de sortida.

Precondició

true

Postcondició

El canal de sortida conté les dades de tots els processadors del cluster.

Definició a la línia 54 del fitxer cluster.cc.

```
54                                  {
55     for (auto it = m.begin(); it != m.end(); ++it){
56         cout « it->first « endl;
57         it->second.write();
58     }
59 }
```

3.1.3.23 write_structure()

```
void cluster::write_structure ( )
```

Escriu l'estructura del cluster al canal de sortida.

Precondició

El cluster està inicialitzat.

Postcondició

L'estructura del cluster es troba al canal de sortida.

Definició a la línia 74 del fitxer cluster.cc.

```
74                                  {
75     write_i_struct(c);
76     cout « endl;
77 }
```

3.1.4 Documentació de les Dades Membre

3.1.4.1 c

```
BinTree<string> cluster::c  [private]
```

Definició a la línia 26 del fitxer cluster.hh.

3.1.4.2 m

```
map<string, Procesor> cluster::m  [private]
```

Definició a la línia 27 del fitxer cluster.hh.

3.1.4.3 time

```
int cluster::time  [private]
```

Definició a la línia 28 del fitxer cluster.hh.

La documentació d'aquesta classe es va generar a partir dels següents fitxers:

- cluster.hh
- cluster.cc

3.2 Referència de l'Estructura cluster::info

Atributs Públics

- bool cap
- int forat
- int freemem
- string idprocesor

3.2.1 Descripció Detallada

Definició a la línia 30 del fitxer cluster.hh.

3.2.2 Documentació de les Dades Membre

3.2.2.1 cap

```
bool cluster::info::cap
```

Definició a la línia 31 del fitxer cluster.hh.

3.2.2.2 forat

```
int cluster::info::forat
```

Definició a la línia 32 del fitxer cluster.hh.

3.2.2.3 freemem

```
int cluster::info::freemem
```

Definició a la línia 33 del fitxer cluster.hh.

3.2.2.4 idprocesor

```
string cluster::info::idprocesor
```

Definició a la línia 34 del fitxer cluster.hh.

La documentació d'aquesta estructura es va generar a partir del següent fitxer:

- cluster.hh

3.3 Referència de l'Estructura proces_area::Priority

Atributs Públics

- int accepted = 0
- int rejected = 0
- list< Proces > proclist

3.3.1 Descripció Detallada

Definició a la línia 26 del fitxer pending_proces_area.hh.

3.3.2 Documentació de les Dades Membre

3.3.2.1 accepted

```
int proces_area::Priority::accepted = 0
```

Definició a la línia 27 del fitxer pending_proces_area.hh.

3.3.2.2 proclist

```
list<Proces> proces_area::Priority::proclist
```

Definició a la línia 29 del fitxer pending_proces_area.hh.

3.3.2.3 rejected

```
int proces_area::Priority::rejected = 0
```

Definició a la línia 28 del fitxer pending_proces_area.hh.

La documentació d'aquesta estructura es va generar a partir del següent fitxer:

- pending_proces_area.hh

3.4 Referència de la Classe Proces

Representa un procès format per un identificador, una prioritat, el temps necessari, l'espai que ocupa a memòria i el temps actual.

```
#include <proces.hh>
```

Mètodes públics

- Proces ()
- Proces (string id, int mem, int necessarytime)
- Proces (string id)
- void read_proces ()

  Llegeix les dades d'un procès del canal d'entrada.
- string identifier ()

  Retorna l'identificador del procès.
- int memory ()

  Retorna la quantitat de memòria que ocupa el procès.
- void write_proces ()

  Escriu les dades del procès al canal de sortida.
- int remainingtime ()

  Retorna el temps que falta per a acabar el procès.
- int increase_time (int t)

  Resta t al temps restant.

Atributs Privats

- string id
- int mem
- int necesarytime
- int timeleft

3.4.1 Descripció Detallada

Representa un procès format per un identificador, una prioritat, el temps necessari, l'espai que ocupa a memòria i el temps actual.

Definició a la línia 22 del fitxer proces.hh.

3.4.2 Documentació del Constructor i el Destructor

3.4.2.1 Proces() [1/3]

```
Proces::Proces ( )
```

Definició a la línia 3 del fitxer proces.cc.

```
3               {
4     id = "0";
5     mem = 0;
6     necesarytime = 0;
7     timeleft = 0;
8 }
```

3.4.2.2 Proces() [2/3]

```
Proces::Proces (
            string id,
            int mem,
            int necessarytime )
```

Definició a la línia 10 del fitxer proces.cc.

```
10                                                  {
11     this->id = id;
12     this->mem = mem;
13     this->necesarytime = necesarytime;
14     timeleft = necesarytime;
15 }
```

3.4.2.3 Proces() [3/3]

```
Proces::Proces (
            string id )
```

Definició a la línia 17 del fitxer proces.cc.

```
17                          {
18     this->id = id;
19 }
```

3.4.3 Documentació de les Funcions Membre

3.4.3.1 identifier()

```
string Proces::identifier ( )
```

Retorna l'identificador del procès.

Precondició

true

Postcondició

Retorna l'identificador del procès

Retorna

string Identificador

Definició a la línia 31 del fitxer proces.cc.

```
31                          {
32     return id;
33 }
```

3.4.3.2 increase_time()

```
int Proces::increase_time (
            int t )
```

Resta t al temps restant.

Precondició

true

Postcondició

Retorna el temps restant

Paràmetres

t	temps que s'ha incrementat

Retorna

int temps restant

Definició a la línia 43 del fitxer proces.cc.

```
43                                  {
44     timeleft -= t;
45     return timeleft;
46 }
```

3.4.3.3 memory()

```
int Proces::memory ( )
```

Retorna la quantitat de memòria que ocupa el procès.

Precondició

true

Postcondició

Retorna memòria que ocupa el procès

Retorna

int memòria

Definició a la línia 39 del fitxer proces.cc.

```
39                  {
40     return mem;
41 }
```

3.4.3.4 read_proces()

```
void Proces::read_proces ( )
```

Llegeix les dades d'un procès del canal d'entrada.

Precondició

true

Postcondició

Les dades introduides es troben al P.I.

Definició a la línia 21 del fitxer proces.cc.

```
21                          {
22     string i;
23     int m, t;
24     cin » i » m » t;
25     id = i;
26     mem = m;
27     necesarytime = t;
28     timeleft = t;
29 }
```

3.4.3.5 remainingtime()

```
int Proces::remainingtime ( )
```

Retorna el temps que falta per a acabar el procès.

Precondició

true

Postcondició

Retorna el temps que falta per a acabar el procès

Retorna

int temps restant

Definició a la línia 47 del fitxer proces.cc.

```
47                          {
48     return timeleft;
49 }
```

3.4.3.6 write_proces()

```
void Proces::write_proces ( )
```

Escriu les dades del procès al canal de sortida.

Precondició

true

Postcondició

Les dades del procès es troben al canal de sortida.

Definició a la línia 35 del fitxer proces.cc.

```
35                          {
36     cout « id « " " « mem « " " « timeleft « endl;
37 }
```

3.4.4 Documentació de les Dades Membre

3.4.4.1 id

```
string Proces::id  [private]
```

Definició a la línia 24 del fitxer proces.hh.

3.4.4.2 mem

```
int Proces::mem  [private]
```

Definició a la línia 25 del fitxer proces.hh.

3.4.4.3 necesarytime

```
int Proces::necesarytime  [private]
```

Definició a la línia 26 del fitxer proces.hh.

3.4.4.4 timeleft

```
int Proces::timeleft  [private]
```

Definició a la línia 27 del fitxer proces.hh.

La documentació d'aquesta classe es va generar a partir dels següents fitxers:

- proces.hh
- proces.cc

3.5 Referència de la Classe proces_area

Representa una area de processos pendents. Formada per una llista de programes ordenada per prioritat.

```
#include <pending_proces_area.hh>
```

Classes

- struct Priority

Mètodes públics

- proces_area (const vector< string > &p)

  Construeix una nova area de procesos pendents.
- bool add_priority (string id)

  Afegeix la prioritat id al conjunt de prioritats.
- int remove_priority (string id)

  Elimina la prioritat id del conjunt de prioritats.
- int add_proces (string id, Proces &p)

  Afegeix un procès a l'area de processos pendents.
- void add_proces_to_cluster (cluster &c, int n)

  Inteta afegir n processos a c.
- bool exist_priority (string id)

  Retorna si l'identificador existeix en el conjunt de prioritats.
- void write_priority (string id)

  Escriu la llista de la prioritat id.
- void write_pending_proces_area ()

  Esctiu l'area de processos pendents al canal de sorida.

Mètodes Privats

- bool search (string idpri, string idproc)

  Retorna si el processador ammb identificador idproc es troba a la prioritat idpri.

Atributs Privats

- map< string, Priority > pr

3.5.1 Descripció Detallada

Representa una area de processos pendents. Formada per una llista de programes ordenada per prioritat.

Definició a la línia 24 del fitxer pending_proces_area.hh.

3.5.2 Documentació del Constructor i el Destructor

3.5.2.1 proces_area()

```
proces_area::proces_area (
            const vector< string > & p )
```

Construeix una nova area de procesos pendents.

Precondició

p es un conjunt no buit d'identificadors de prioritat.

Postcondició

S'ha creat l'objecte amb p al parametre implicit.

Paràmetres

p	

Definició a la línia 14 del fitxer pending_proces_area.cc.

```
14                                                  {
15     int n = p.size();
16     for (int i=0; i<n; ++i){
17         pr[p[i]];
18     }
19 }
```

3.5.3 Documentació de les Funcions Membre

3.5.3.1 add_priority()

```
bool proces_area::add_priority (
            string id )
```

Afegeix la prioritat id al conjunt de prioritats.

Precondició

true

Postcondició

S'afegeix la prioritat id al conjunt de prioritats.

Paràmetres

id	Idenificador de prioritat

Retorna

true si l'identificador s'ha afegit a la llista de prioritats

false si l'identificador no es pot afegir a la llista de prioritats

Definició a la línia 21 del fitxer pending_proces_area.cc.

```
21                                              {
22     if(pr.find(id) == pr.end()){
23         pr[id];
24         return true;
25     }
26     return false;
27 }
```

3.5.3.2 add_proces()

```
int proces_area::add_proces (
              string id,
              Proces & p )
```

Afegeix un procès a l'area de processos pendents.

Precondició

true

Postcondició

Retorna l'area de processos pendents amb els processos de prioritat id eliminats.

Paràmetres

p	Procès que s'ha d'afegir a l'àrea
id	Identificador de prioritat

Retorna

true si el procès s'ha afegit a la llista de prioritats

false si el procès no es pot afegir a la llista de prioritats

Definició a la línia 36 del fitxer pending_proces_area.cc.

```
36                                                  {
37     if(pr.find(id) == pr.end()){
38         return 1;
39     }
40     if(not search(id, p.identifier())){
41         pr[id].proclist.push_back(p);
42         return 0;
43     }
44     return 2;
45 }
```

3.5.3.3 add_proces_to_cluster()

```
void proces_area::add_proces_to_cluster (
              cluster & c,
              int n )
```

Inteta afegir n processos a c.

Precondició

true

Postcondició

El cluster c conté un nombere <= n processos nous

Paràmetres

c	Cluster on s'hann d'afegir els processos
n	Nombre de processos que s'han d'intentar afegir

Definició a la línia 47 del fitxer pending_proces_area.cc.

```
47                                                                  {
48     if (n == 0) return;
49     for (auto itmap = pr.begin(); itmap != pr.end(); ++itmap){
50      auto itlist = itmap->second.proclist.begin();
51      while (itlist != itmap->second.proclist.end()){
52          bool erased = false;
53          Proces p = *itlist;
54          string id = c.where_add(p.identifier(), p.memory());
55          if (id == "..."){
56              ++ itmap->second.rejected;
57          }
58          else {
59              ++itmap->second.accepted;
60              c.add_proces_directly(id, p);
61              itlist = itmap->second.proclist.erase(itlist);
62              erased = true;
63          }
64          if (not erased) ++itlist;
65          --n;
66          if (n == 0) return;
67      }
68     }
```

```
69 }
```

3.5.3.4 exist_priority()

```
bool proces_area::exist_priority (
            string id )
```

Retorna si l'identificador existeix en el conjunt de prioritats.

Precondició

true

Postcondició

Retorna si l'identificador existeix en el conjunt de prioritats.

Paràmetres

id	Identificador de prioritat.

Retorna

true si l'identificador s'ha afegit a la llista de prioritats.

false si l'identificador no es pot afegir a la llista de prioritats.

Definició a la línia 71 del fitxer pending_proces_area.cc.

```
71                                              {
72     if(pr.find(id) == pr.end()) return false;
73     return true;
74 }
```

3.5.3.5 remove_priority()

```
int proces_area::remove_priority (
            string id )
```

Elimina la prioritat id del conjunt de prioritats.

Precondició

true

Postcondició

Retorna l'area de processos pendents amb els processos de prioritat id eliminats.

Paràmetres

id	identificador de prioritats

Retorna

true si l'identificador s'ha esborrat de la llista de prioritats.

false si l'identificador no es pot esborrar de la llista de prioritats.

Definició a la línia 29 del fitxer pending_proces_area.cc.

```
29                                          {
30     if(pr.find(id) == pr.end()) return 1;
31     if ( not pr[id].proclist.empty()) return 2;
32     pr.erase(pr.find(id));
33     return 0;
34 }
```

3.5.3.6 search()

```
bool proces_area::search (
              string idpri,
              string idproc )  [private]
```

Retorna si el processador ammb identificador idproc es troba a la prioritat idpri.

Paràmetres

idpri	Identificador de prioritat
idproc	Identificador del processador

Retorna

true El processador està a la prioritat

false El processador no està a la prioritat

Definició a la línia 3 del fitxer pending_proces_area.cc.

```
3                                                   {
4     list<Proces> l = pr[idpri].proclist;
5     for(auto it = l.begin(); it != l.end(); ++it){
6         Proces p = *it;
7         if (p.identifier()==idproc){
8             return true;
9         }
10     }
11     return false;
12 }
```

3.5.3.7 write_pending_proces_area()

```
void proces_area::write_pending_proces_area ( )
```

Esctiu l'area de processos pendents al canal de sorida.

Precondició

L'area de processos ha d'estar inicialitda.

Postcondició

S'escriu la llista de procesos al canal de sortida.

Definició a la línia 85 del fitxer pending_proces_area.cc.

```
85                                          {
86      for (auto i = pr.begin(); i != pr.end(); i++){
87         cout « i->first « endl;
88         list<Proces> l = i->second.proclist;
89          for(auto it = l.begin(); it != l.end(); ++it){
90             Proces p = *it;
91             p.write_proces();
92         }
93         cout «  i->second.accepted « " " «  i->second.rejected « endl;
94      }
95 }
```

3.5.3.8 write_priority()

```
void proces_area::write_priority (
              string id )
```

Escriu la llista de la prioritat id.

Precondició

true

Postcondició

La llista de prioritats es troba al canal de sortida.

Paràmetres

id	Identificador de prioritat.

Definició a la línia 76 del fitxer pending_proces_area.cc.

```
76                                          {
77     list<Proces> l = pr[id].proclist;
78     for(auto it = l.begin(); it != l.end(); ++it){
79         Proces p = *it;
80         p.write_proces();
81     }
82     cout « pr[id].accepted « " " « pr[id].rejected « endl;
83 }
```

3.5.4 Documentació de les Dades Membre

3.5.4.1 pr

```
map<string,Priority> proces_area::pr  [private]
```

Definició a la línia 31 del fitxer pending_proces_area.hh.

La documentació d'aquesta classe es va generar a partir dels següents fitxers:

- pending_proces_area.hh
- pending_proces_area.cc

3.6 Referència de la Classe Procesor

Representa un processador. Format per una identificació, una memòria i una llista de processos.

```
#include <procesor.hh>
```

Mètodes públics

- Procesor ()
- Procesor (string id, int maxmem)
- void read_procesor ()

  Llegeix les dades del processador del sistema d'entrada.
- void compact_memory ()

  Compacta la memoria del processador.
- bool is_executing ()

  Retorna si s'esta executant algun proces.
- string identifier ()

  Retorna l'identificador del processador.
- void write ()

  Escriu les dades del processdor al canal de sortida.
- int maxmem ()
- int add_proces (Proces &p)

  Afegeix el rocès p al processador.
- void increase_time (int t)
- bool remove_proces (string idproces)
- pair< int, int > fit (int mem)
- bool exist (string identifier)

Mètodes Privats

- void unify_free_memory ()

  Unifica els espais buits de la memòria del processador.

Atributs Privats

- string id
- int max_mem
- int freemem
- map< string, int > m1
- map< int, Proces > m2
- map< int, set< int > > m3

3.6.1 Descripció Detallada

Representa un processador. Format per una identificació, una memòria i una llista de processos.

Definició a la línia 27 del fitxer procesor.hh.

3.6.2 Documentació del Constructor i el Destructor

3.6.2.1 Procesor() [1/2]

```
Procesor::Procesor ( )
```

Definició a la línia 3 del fitxer procesor.cc.

```
3                   {
4   id = "0";
5   max_mem = 0;
6 }
```

3.6.2.2 Procesor() [2/2]

```
Procesor::Procesor (
             string id,
             int maxmem )
```

Definició a la línia 8 del fitxer procesor.cc.

```
8                                           {
9   this->id = id;
10   max_mem = maxmem;
11   freemem = maxmem;
12   m3[maxmem].insert(0);
13 }
```

3.6.3 Documentació de les Funcions Membre

3.6.3.1 add_proces()

```
int Procesor::add_proces (
            Proces & p )
```

Afegeix el rocès p al processador.

Precondició

El processador no contè cap proces amb el mateix identificador que p

Postcondició

Retorna la memòria maxima del processador

Paràmetres

p	

Retorna

int

Definició a la línia 114 del fitxer procesor.cc.

```
114                                 {
115   int necesary_mem = p.memory();
116   if (m1.find(p.identifier()) != m1.end()) return 2;
117   if (m3.lower_bound(necesary_mem) == m3.end()) return 3;
118   auto it = m3.lower_bound(necesary_mem);
119   int pos = *(it->second.begin());
120   int memory = it->first;
121   if (memory > necesary_mem){
122     m3[memory-necesary_mem].insert(pos+necesary_mem);
123   }
124   m3[memory].erase(pos);
125   if(m3[memory].empty()) m3.erase(memory);
126   m1[p.identifier()] = pos;
127   m2[pos] = p;
128   freemem -= necesary_mem;
129   return 0;
130 }
```

3.6.3.2 compact_memory()

```
void Procesor::compact_memory ( )
```

Compacta la memoria del processador.

Precondició

true

Postcondició

Es retorna el mateix processador amb la memòria compactada.

Definició a la línia 68 del fitxer procesor.cc.

```
68                                {
69   if (m2.empty() or m3.empty()) return;
70   map <int, Proces> m2aux;
71   int pos = 0;
72   for(auto it = m2.begin(); it != m2.end(); ++it){
73     m1[it->second.identifier()] = pos;
74     m2aux[pos]= it->second;
75     pos += it->second.memory();
76   }
77   m3[max_mem-pos+1].insert(max_mem-pos+1);
78   m2 = m2aux;
79 }
```

3.6.3.3 exist()

```
bool Procesor::exist (
              string identifier )
```

Definició a la línia 162 del fitxer procesor.cc.

```
162                                        {
163   if (m1.find(identifier) == m1.end()) return false;
164   return true;
165 }
```

3.6.3.4 fit()

```
pair< int, int > Procesor::fit (
              int mem )
```

Definició a la línia 149 del fitxer procesor.cc.

```
149                                       {
150   pair <int, int> out;
151   auto it = m3.lower_bound(mem);
152   if (it == m3.end()){
153     out.first = -1;
154     out.second = -1;
155     return out;
156   }
157   out.first = it-> first - mem;
158   out.second = freemem - mem;
159   return out;
160 }
```

3.6.3.5 identifier()

```
string Procesor::identifier ( )
```

Retorna l'identificador del processador.

Precondició

true

Postcondició

Retorna l'identificador del processador.

Definició a la línia 85 del fitxer procesor.cc.

```
85                              {
86   return id;
87 }
```

3.6.3.6 increase_time()

```
void Procesor::increase_time (
              int t )
```

Definició a la línia 95 del fitxer procesor.cc.

```
95                                  { // definir un map = m2 aux per a no editar m2 durant l'execució
96   auto it = m2.begin();
97   while (it != m2.end()){
98     bool erased = false;
99     if (m2[it->first].increase_time(t) <= 0){
100       int inipos = it->first;
101       int memory = it->second.memory();
102       string idproces = it->second.identifier();
103       m1.erase(idproces);
104       it = m2.erase(it);
105       m3[memory].insert(inipos);
106       unify_free_memory();
107       erased = true;
108       freemem = freemem + memory;
109     }
110     if (not erased) ++it;
111   }
112 }
```

3.6.3.7 is_executing()

```
bool Procesor::is_executing ( )
```

Retorna si s'esta executant algun proces.

Precondició

true

Postcondició

Retorna si s'esta executant algun procès.

Retorna

true si s'età executant algún procès.

false si no s'està executant cap procès.

Definició a la línia 81 del fitxer procesor.cc.

```
81                                  {
82   return not m1.empty();
83 }
```

3.6.3.8 maxmem()

```
int Procesor::maxmem ( )
```

Precondició

true

Postcondició

Retorna la memòria maxima del processador

Definició a la línia 132 del fitxer procesor.cc.

```
132                     {
133   return max_mem;
134 }
```

3.6.3.9 read_procesor()

```
void Procesor::read_procesor ( )
```

Llegeix les dades del processador del sistema d'entrada.

Precondició

true

Postcondició

Les dades del sistema d'entrada estan desades al P.I.

3.6.3.10 remove_proces()

```
bool Procesor::remove_proces (
            string idproces )
```

Definició a la línia 136 del fitxer procesor.cc.

```
136                                         {
137   if (m1.find(idproces) == m1.end()) return false;
138   auto it = m1.find(idproces);
139   int inipos = it->second;
140   int memory = m2[inipos].memory();
141   m1.erase(idproces);
142   m2.erase(inipos);
143   m3[memory].insert(inipos);
144   freemem += memory;
145   unify_free_memory();
146   return true;
147 }
```

3.6.3.11 unify_free_memory()

```
void Procesor::unify_free_memory ( )  [private]
```

Unifica els espais buits de la memòria del processador.

Definició a la línia 15 del fitxer procesor.cc.

```
15                                {
16     m3.clear();
17     int posicioActual = 0;
18      for (auto it = m2.begin(); it != m2.end(); ++it) {
19          int posicio = it->first;
20          int espai = it->second.memory();
21
22          if (posicio > posicioActual) {
23              int espaiLliure = posicio - posicioActual;
24              m3[espaiLliure].insert(posicioActual);
25          }
26
27          posicioActual = posicio + espai;
28      }
29
30      // Comprova l'espai lliure després de l'última entrada
31      int ultimaPosicio = posicioActual;
32      if (ultimaPosicio < max_mem) {
33          int espaiLliure = max_mem - ultimaPosicio;
34          m3[espaiLliure].insert(ultimaPosicio);
35      }
36 }
```

3.6.3.12 write()

```
void Procesor::write ( )
```

Escriu les dades del processdor al canal de sortida.

Precondició

El procesador està inicialitzat.

Postcondició

El canal de sortida conté les dades del procesador.

Definició a la línia 89 del fitxer procesor.cc.

```
89                        {
90    for (auto it = m2.begin(); it != m2.end(); ++it){
91      cout « it->first « " " « it->second.identifier() « " " « it->second.memory() « " " «
        it->second.remainingtime() « endl;
92    }
93 }
```

3.6.4 Documentació de les Dades Membre

3.6.4.1 freemem

```
int Procesor::freemem  [private]
```

Definició a la línia 31 del fitxer procesor.hh.

3.6.4.2 id

```
string Procesor::id  [private]
```

Definició a la línia 29 del fitxer procesor.hh.

3.6.4.3 m1

```
map<string, int> Procesor::m1  [private]
```

Definició a la línia 32 del fitxer procesor.hh.

3.6.4.4 m2

```
map<int, Proces> Procesor::m2  [private]
```

Definició a la línia 33 del fitxer procesor.hh.

3.6.4.5 m3

```
map<int, set<int> > Procesor::m3  [private]
```

Definició a la línia 34 del fitxer procesor.hh.

3.6.4.6 max_mem

```
int Procesor::max_mem  [private]
```

Definició a la línia 30 del fitxer procesor.hh.

La documentació d'aquesta classe es va generar a partir dels següents fitxers:

- procesor.hh
- procesor.cc

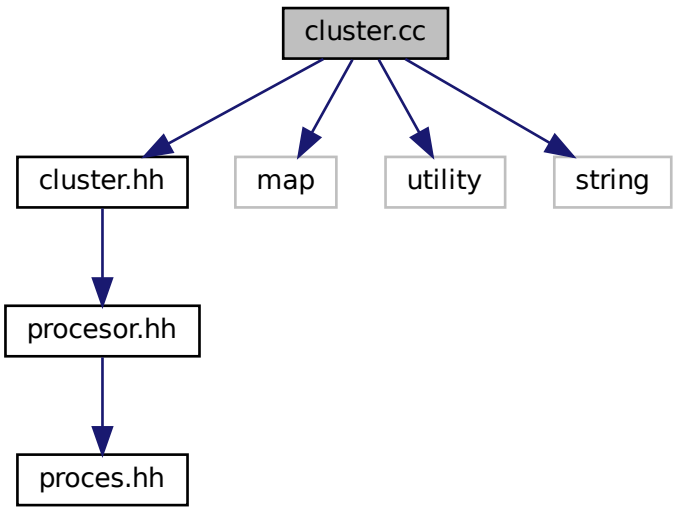
Chapter 4

Documentació dels Fitxers

4.1 Referència del Fitxer cluster.cc

```
#include "cluster.hh"
#include <map>
#include <utility>
#include <string>
```

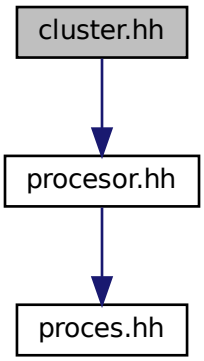
Inclou el graf de dependències per a cluster.cc:



4.2 Referència del Fitxer cluster.hh

Especificació de la classe cluster.

```
#include "procesor.hh"
```
Inclou el graf de dependències per a cluster.hh:



Classes

- class cluster

  Representa un cluster en forma d'arbre binari. Està format per processadors i processos.
- struct cluster::info

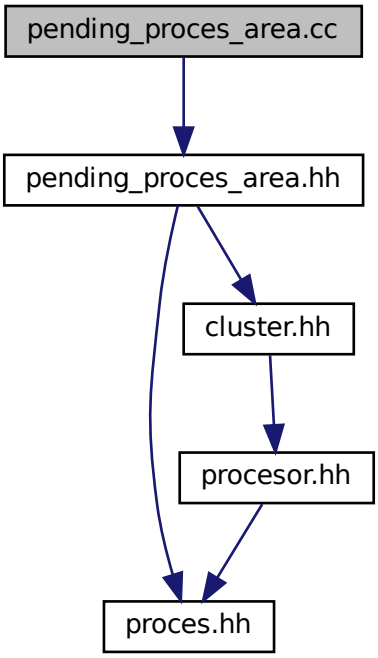
4.2.1 Descripció Detallada

Especificació de la classe cluster.

4.3 Referència del Fitxer pending_proces_area.cc

```
#include "pending_proces_area.hh"
```

Inclou el graf de dependències per a pending_proces_area.cc:

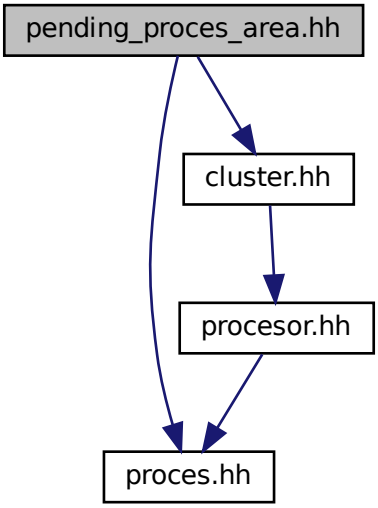


4.4 Referència del Fitxer pending_proces_area.hh

Especificació de l'area de processos pendents.

```
#include "proces.hh"
#include "cluster.hh"
```

Inclou el graf de dependències per a pending_proces_area.hh:



Classes

- class proces_area

  Representa una area de processos pendents. Formada per una llista de programes ordenada per prioritat.
- struct proces_area::Priority

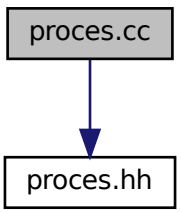
4.4.1 Descripció Detallada

Especificació de l'area de processos pendents.

4.5 Referència del Fitxer proces.cc

```
#include "proces.hh"
```

Inclou el graf de dependències per a proces.cc:



4.6 Referència del Fitxer proces.hh

Esprcificació de la classe procès.

Classes

- class Proces

  Representa un procès format per un identificador, una prioritat, el temps necessari, l'espai que ocupa a memòria i el temps actual.

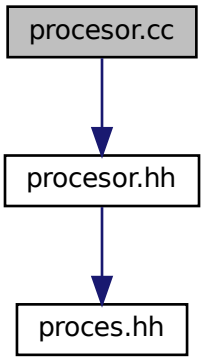
4.6.1 Descripció Detallada

Esprcificació de la classe procès.

4.7 Referència del Fitxer procesor.cc

```
#include "procesor.hh"
```

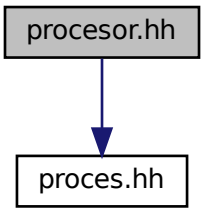
Inclou el graf de dependències per a procesor.cc:



4.8 Referència del Fitxer procesor.hh

Especificació de la classe processador.

```
#include "proces.hh"
```
Inclou el graf de dependències per a procesor.hh:



Classes

- class Procesor

  Representa un processador. Format per una identificació, una memòria i una llista de processos.

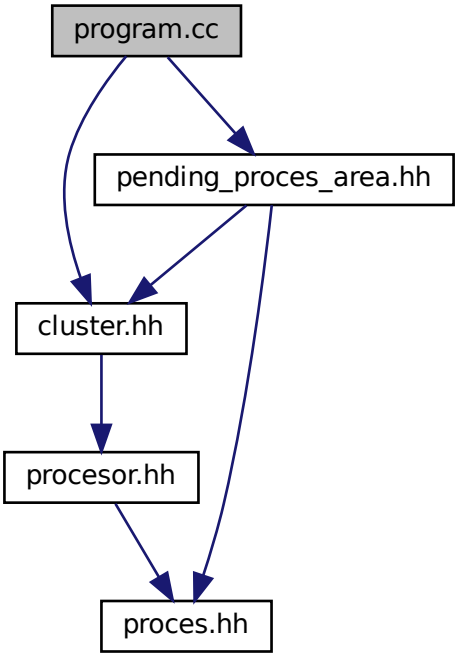
4.8.1 Descripció Detallada

Especificació de la classe processador.

4.9 Referència del Fitxer program.cc

```
#include "cluster.hh"
#include "pending_proces_area.hh"
```

Inclou el graf de dependències per a program.cc:



Funcions

- int main ()

4.9.1 Documentació de les Funcions

4.9.1.1 main()

```
int main ( )
```

Definició a la línia 22 del fitxer program.cc.

```
22           {
23   cluster c;
24   c.read_cluster();
25   // Llegim la llista de prioritats
26   int n;
27   cin » n;
28   vector<string> priorities (n);
29   for (int i = 0; i<n; ++i){
30     string in;
31     cin » in;
32     priorities[i]=in;
33   }
34   proces_area a (priorities);
35   // Bucle de comandes
```

```
36   string comand;
37   cin » comand;
38   while (comand != "fin"){
39     if(comand == "configurar_cluster" or comand == "cc"){
40       cout « "#" « comand « endl;
41       cluster copy;
42       copy.read_cluster();
43       c = copy;
44     }
45     else if(comand == "modificar_cluster" or comand == "mc"){
46       string id;
47       cin » id;
48       cout « "#" « comand « " " « id « endl;
49       cluster n;
50       n.read_cluster();
51       if (c.exist(id)){
52         Procesor p = c.search_procesor(id);
53         if(not p.is_executing()){
54              if (c.aux_procesors(id)){
55                 c.insert_cluster(n.map_cluster(), n.bintree_cluster(), id);
56              }
57              else cout « "error: procesador con auxiliares" « endl;
58         }
59         else cout « "error: procesador con procesos" « endl;
60       }
61       else cout « "error: no existe procesador" « endl;
62     }
63     else if(comand == "alta_prioridad" or comand == "ap"){
64       string id;
65       cin » id;
66       cout « "#" « comand « " " « id « endl;
67       if(not a.add_priority(id)) cout « "error: ya existe prioridad" « endl;
68     }
69     else if(comand == "baja_prioridad" or comand == "bp"){
70       string id;
71       cin » id;
72       cout « "#" « comand « " " « id « endl;
73       int error = a.remove_priority(id);
74       if (error == 1) cout « "error: no existe prioridad" « endl;
75       else if (error == 2) cout « "error: prioridad con procesos" « endl;
76     }
77     else if(comand == "alta_proceso_espera" or comand == "ape"){
78       string id;
79       cin » id;
80       Proces p;
81       p.read_proces();
82       cout « "#" « comand « " " « id « " " « p.identifier() « endl;
83       int error = a.add_proces(id, p);
84       if (error == 1) cout « "error: no existe prioridad" « endl;
85       else if (error == 2) cout « "error: ya existe proceso" « endl;
86     }
87     else if(comand == "alta_proceso_procesador" or comand == "app"){
88       string idprocesor;
89       cin » idprocesor;
90       Proces p;
91       p.read_proces();
92       cout « "#" « comand « " " « idprocesor « " " « p.identifier() « endl;
93       int error = c.add_proces_directly(idprocesor, p);
94       if (error == 1) cout « "error: no existe procesador" « endl;
95       else if (error == 2) cout « "error: ya existe proceso" « endl;
96       else if (error == 3) cout « "error: no cabe proceso" « endl;
97     }
98     else if(comand == "baja_proceso_procesador" or comand == "bpp"){
99       string idprocesor, idproces;
100       cin » idprocesor » idproces;
101       cout « "#" « comand « " " « idprocesor « " " « idproces « endl;
102       int error = c.remove_proces(idprocesor, idproces);
103       if (error == 1) cout « "error: no existe procesador" « endl;
104       else if (error == 2) cout « "error: no existe proceso" « endl;
105
106       if (not c.remove_proces(idprocesor, idproces)) cout « error « endl;
107     }
108     else if(comand == "enviar_procesos_cluster" or comand == "epc"){
109       int n;
110       cin » n;
111       cout « "#" « comand « " " « n « endl;
112       a.add_proces_to_cluster(c, n);
113     }
114     else if(comand == "avanzar_tiempo" or comand == "at"){
115       int t;
116       cin » t;
117       cout « "#" « comand « " " « t « endl;
118       c.increase_time(t);
119     }
120     else if(comand == "imprimir_prioridad" or comand == "ipri"){
121       string id;
122       cin » id;
```

```
123       cout « "#" « comand « " " « id « endl;
124       if (a.exist_priority(id)) a.write_priority(id);
125       else cout « "error: no existe prioridad" « endl;
126     }
127     else if(comand == "imprimir_area_espera" or comand == "iae"){
128       cout « "#" « comand « endl;
129       a.write_pending_proces_area();
130     }
131     else if(comand == "imprimir_procesador" or comand == "ipro"){
132       string id;
133       cin » id;
134       cout « "#" « comand « " " « id « endl;
135       if (c.exist(id)){
136         Procesor pro = c.search_procesor(id);
137         pro.write();
138       }
139       else cout « "error: no existe procesador" « endl;
140     }
141     else if(comand == "imprimir_procesadores_cluster" or comand == "ipc"){
142       cout « "#" « comand « endl;
143       c.write_procesors();
144     }
145     else if(comand == "imprimir_estructura_cluster" or comand == "iec"){
146       cout « "#" « comand « endl;
147       c.write_structure();
148     }
149     else if(comand == "compactar_memoria_procesador" or comand == "cmp"){
150       string id;
151       cin » id;
152       cout « "#" « comand « " " « id « endl;
153       if(c.exist(id)){
154         Procesor pro = c.search_procesor(id);
155         pro.compact_memory();
156         c.modify_processor(id, pro);
157       }
158       else cout « "error: no existe procesador" « endl;;
159     }
160     else if(comand == "compactar_memoria_cluster" or comand == "cmc"){
161       cout « "#" « comand « endl;
162       c.compact_memory();
163     }
164     cin » comand;
165   }
166 }
```