Sistemes Operatius II

Pràctica 2

Vicent Roig / Igor Dzinka

26/10/2014

1. Lectura del fitxer de configuració

Es proposa implementar la lectura del fitxer de configuració. Quina funció fareu servir per llegir aquest fitxer ? La funció fgetc, fgets, fscanf, o fread ?

Com que el fitxer de configuració ve donat amb un format molt determinat (a cada línia es troba una ruta al fitxer), seria igual d'útil utilitzar tant la funció fscanf() com la funció fgets(). Encara que no es dona el cas, si el nom del path o fitxer a llegir tinguessin espai en blanc, la funció fscanf() no retornaria la ruta correctament.

No hem vist útil fer servir la funció fgetc() donat que s'hauria de construir la string de la ruta i nom de fitxer manualment, quan realment el format del fitxer es propici d'aprofitar-se.

El mateix passa amb la funció fread() amb la qual hauríem de fer lo mateix que amb la funció fgetc() amb la diferencia de que abans carregaríem tot el fitxer a la memòria.

Per tant, hem optat per la funció fgets(), que permet llegir el contingut de tota una línia, que en aquest cas es la pròpia ruta de cada fitxer.

2. Estructuració Local

Com afecta el valor de SIZE a hash.c al rendiment del programa?

```
HASHSIZE = 100000
simorgh@simorgh-530U3C:~/SO2/p2/src$ time ./practica2 ../base_dades/llista.cfg
1> ../proves/log.txt

       real    0m42.487s
       user    0m41.738s
       sys     0m0.699s

HASHSIZE = 1000
simorgh@simorgh-530U3C:~/SO2/p2/src$ time ./practica2 ../base_dades/llista.cfg
1> ../proves/log.txt

       real    0m59.934s
       user    0m59.176s
       sys     0m0.683s

HASHSIZE = 100
simorgh@simorgh-530U3C:~/SO2/p2/src$ time ./practica2 ../base_dades/llista.cfg
1> ../proves/log.txt

       real    3m26.755s
       user    3m25.777s
       sys     0m0.791s
```

Es tracta de la mida de hashtable i afecta directament al temps d'estructuració local, podem observar realitzant diverses proves, que en general, per un valor inferior augmenta el temps de còmput, i en contraposició, per valors més grans el temps necessari per processar-ho disminueix.

Cal a dir que això es significatiu fins a cert punt, donat que una mida determinada per la hashtable millorarà o empitjorarà el temps de càlcul en funció de les paraules a processar i en

tots casos trobem valors que arriben a un compromís tamany-eficiéncia, on ja no compensa augmentar aquest valor. Establir una mida molt gran per la taula hash ocupa espai a memòria i la seva repercussió no és proporcional a les entrades. Cal trobar un equilibri.

Perquè ?

Observem el següent fragment de codi:

```
/**
* Find item containing the specified primary_key. Returns the data
 * that it points to (not the item itself).
 *
 */
ListData *findList(List *l, TYPE_LIST_PRIMARY_KEY primary_key){
        ListItem *current;

        current = l->first;
        while (current != NULL){
                if (compEQ(current->data->primary_key, primary_key)) return (current->data);
                current = current->next;
        }
        return (NULL);
}
```

Tenim que pensar que, quantes menys entrades tingui la taula hash, més elements tindrà la corresponent llista enllaçada associada a aquella posició, donat que si introduïm un gran nombre de paraules es tindran que repartir en funció del seu hashcode a la posició corresponen.

Per tant, el temps d'accés del recorregut per llista que es realitza tant en la cerca (*per determinar si una paraula existeix o no a una posició de la taula)*, com al cas de la inserció *(recordem que al tractar-se d'una llista enllaçada tenim que recórrer aquesta fins al final)* augmentarà, ja que el procés resulta molt més costós en proporció.

Per altra banda, si establim un tamany de hashtable massa gran per un volum petit de dades, quedaran moltes llistes enllaçades amb 0 elements i estirem ocupant innecessàriament memòria (entrades a la hash) que realment tenen llistes que no son utilitzades.

A part de la mida de la taula hash s'ha de definir una funció hash que ens asseguri que distribueix equitativament les dades a aquesta taula, evitant una hashtable excessivament desequilibrada.

El principal problema de la funció hash que utilitzem a la pràctica es que assignarà el mateix *hashcode* a P. Ex. "hola" i "halo".

Probablement afegint operacions no commutatives (evitant el sumatori i producte) i introduint algun pes posicional en funció del índex, podríem trobar una funció hash que funcionés millor. Encara i així una bona funció hash sempre dependrà del tipus de dades a analitzar.

3. Resultats

Quina és la paraula més llarga que apareix en els fitxers de text ?

Hem aprofitat el log.txt generat per tractar-ho ràpidament amb parell d'instruccions de python al propi terminal i així obtenir tota la informació aquí sol·licitada.

Aquests càlculs es podrien haver realitzat perfectament a nivell intern amb el llenguatge C.

```
@python
--------------------------------------------------------------------------------
>>> lines = [ data_list.append( tuple( line.rstrip('\n').split(',')) ) for line in
open('log.txt') ]
>>> print max([ (len(i[1]), i[1], i[2]) for i in data_list ])
(69, 'nationalgymnasiummuseumsanatoriumandsuspensoriumsordinaryprivatdocent', 166)
--------------------------------------------------------------------------------
```

Quina paraula és ?

Com podem comprovar, la paraula més llarga obtinguda a l'estructura global és:
nationalgymnasiummuseumsanatoriumandsuspensoriumsordinaryprivatdocent

En quin fitxer apareix ?

Hem modificat temporalment l'output a log.txt per inserir a cada tupla, com informació addicional, l'índex del fitxer. En aquest cas comprovem que es tracta del fitxer referenciat a la línia 167 del fitxer de configuració llista.cfg:

```
index 166 -> (fitxer 167 a llista.cfg) -> etext03/ulyss12.txt -> línia 15040
```

Quantes paraules diferents hi ha en total ?

Per tal de respondre aquesta qüestió simplement hem visualitzat per pantalla el numero de nodes de l'arbre abans de generar el log i alliberar la memòria de l'estructura global.

Obtenim el següent:

```
simorgh@simorgh-530U3C:~/SO2/p2/src$./practica2 ../base_dades/llista.cfg
Paraules diferents: 467961
```