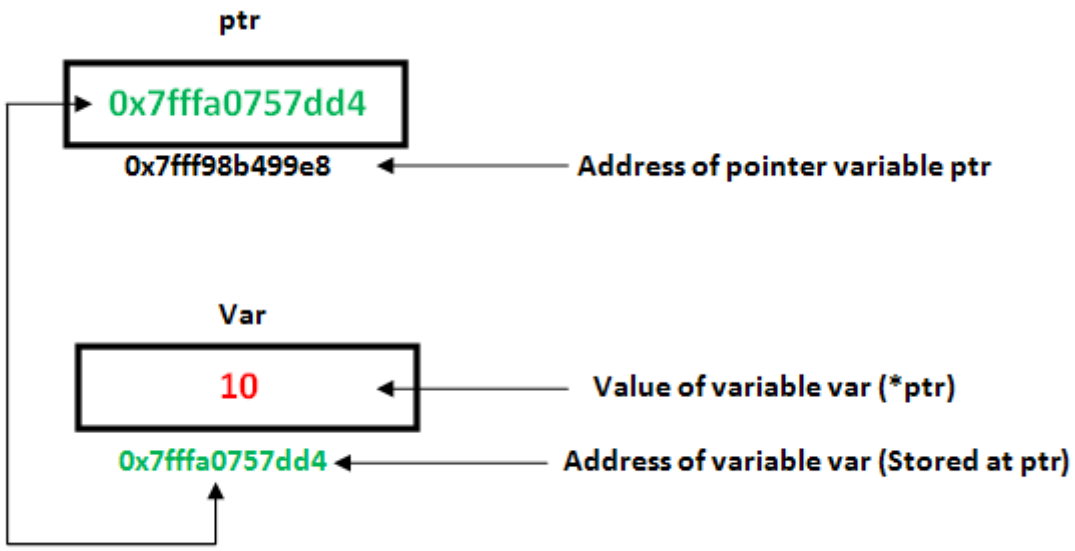
Sistemes Operatius II

Pràctica 1

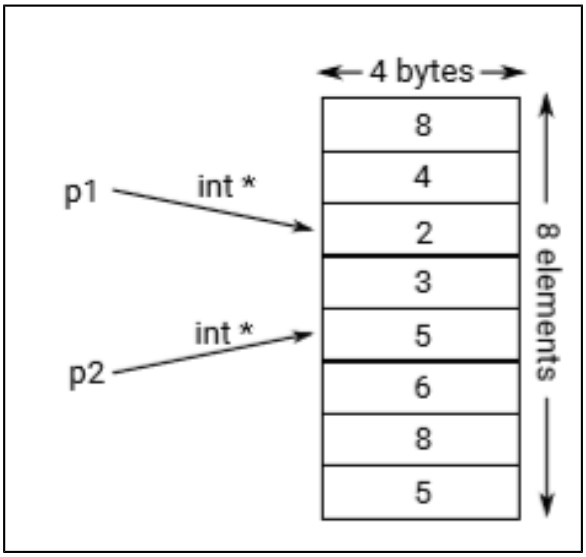
Punters a dades fent servir l'aplicació Quicksort



Xavier de Juan Pulido
David de la Osa Bañales

Aspectes a destacar:

a) **Explicar, ajudant-se d'un dibuix, de la organització de la informació pel cas de les dades de tipus sencers, veure figura 3, i per què cal accedir a les dades tal com s'indica a les línies 8 i 9 del codi 2**



Per obtenira les dades primerament hem d'obrir el fitxer que toca (en aquest cas el fitxer integers.txt), fer un cop *fgets* per obtenir el contingut de la primera línia i fer *atoi* per tal de saber quantes línies hem de llegir.

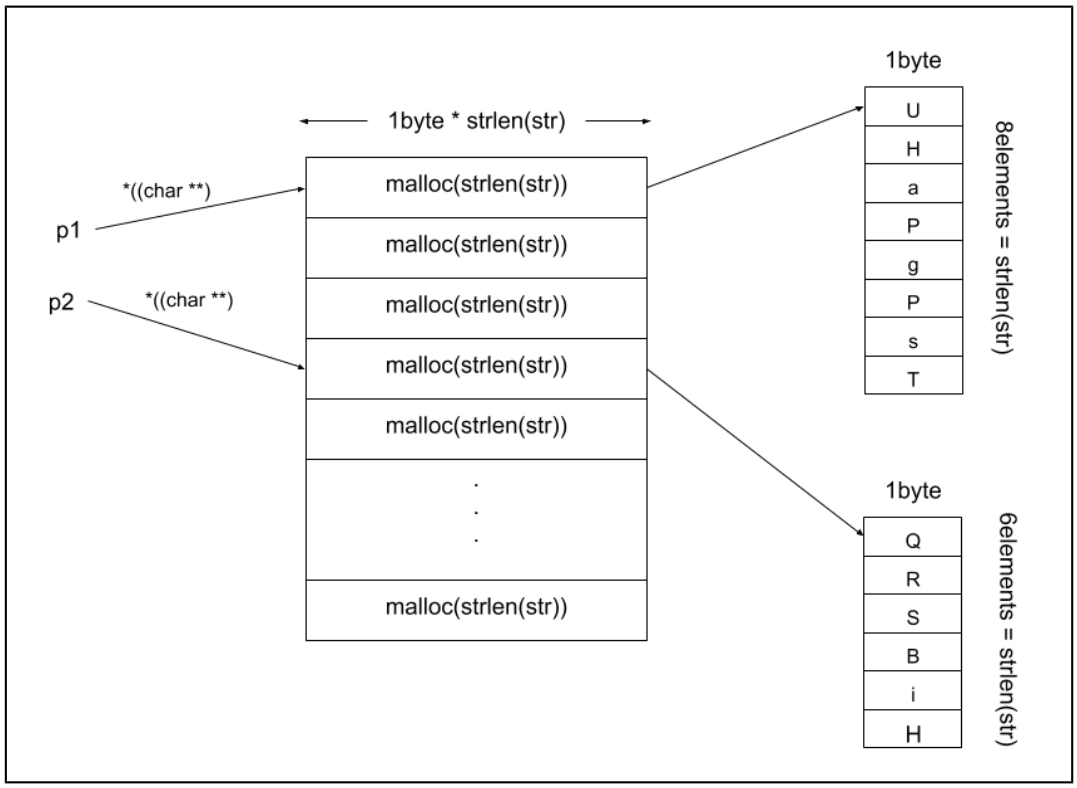
Ara que sabem quants sencers hi ha en el document (10000 sencers en intergers.txt) i sabent que un sencer ocupa 4 bytes podem fer un *malloc* de 10000 elements de 4Bytes cadascun. Per anar-los ficant fem un *for* de 10000 elements on recollim amb *fgets* i *atoi* un sencer i l'afegim en una de les posicions reservades pel *malloc*.

Per acabar, després d'acabar de guardar l'últim sencer, apliquem *qsort*, *printf* i alliberem (*free*) el *int** que hem usat per recórrer les dades.

```
num1 = (int *) p1;
num2 = (int *) p2;
```

Cal accedir a les dades com es mostra en la imatge perquè la funció compara rep dos punters genèrics (*void**) p1, p2 dels quals no se sap a quin tipus de dades apunten. Com nosaltres si que sabem que apunten a sencers fem un cast a *int**.

b) Explicar, ajudant-se d'un dibuix, de la organització de la informació pel cas de les cadenes de caràcters i per què cal accedir a les dades tal com s'indica a les línies 10 i 11 del codi 5.



Veiem aquí la representació del vector de dades en el cas de la cadena de caràcters.

En primer lloc, reservem un espai de memòria de tants elements com hi ha al fitxer (10000 elements), el quadre gran, i, per cada posició d'aquest vector, reservem un espai de memòria igual al nombre de caràcters del 'string' que introduirem en la posició actual.

D'aquesta manera, ens queda una matriu bidimensional de 10000 elements * longitud de cada string.

Per tal d'accedir a cada element (string) del nostre vector ho farem d'aquesta manera:

```
str1 = *((char **) p1);
str2 = *((char **) p2);
```

Accedim d'aquesta forma a les dades perquè nosaltres volem un punter que apunti al string de la posició corresponent, és a dir, el que fem així és assignar al nostre punter l'array de char's (l'string) que hi ha dins del vector principal.

Amb el primer * el que fem és apuntar a una posició del vector principal, amb el segon * apuntem a la primera posició del vector de char's i amb el tercer * accedim a tots els caràcters de l'array de char's, accedint d'aquesta manera a l'string d'una posició del vector principal.

Resultats d’experiments amb Valgrind:

a) Quin missatge us dóna valgrind si no allibereu la memòria de les dades que s’han llegit del fitxer? Podeu indicar com aconseguiu saber a quines línies del codi s’ha reservat la memòria per llegir les dades?

L’output de valgrind al no alliberar la memòria del vector on guardem les dades és aquest:

```
HEAP SUMMARY:
    in use at exit: 40,000 bytes in 1 blocks
    total heap usage: 5 allocs, 4 frees, 88,744 bytes allocated

LEAK SUMMARY:
    definitely lost: 40,000 bytes in 1 blocks
    indirectly lost: 0 bytes in 0 blocks
    possibly lost: 0 bytes in 0 blocks
    still reachable: 0 bytes in 0 blocks
    suppressed: 0 bytes in 0 blocks
```

Això significa que no hem alliberat les 10,000 posicions * 4 (tamany integer) del vector on guardàvem les dades llegides. Al Leak summary, ens apareix aquesta memòria com a ‘definitely lost’ , perquè quan acaba el programa la memòria d’aquest programa es perd quan la pila del main es destrueix, en fer return(0); acabant amb exit(0) aquesta memòria apareixeria com ‘still reachable’.

Afegint el flag –leak-check=full al còrrer el nostre programa amb valgrind, ens permet saber en quines línies del codi es reserva la memòria per llegir les dades.

b) Podeu aconseguir que valgrind us doni un missatge d’error d’accés a memòria? Com?

Sí, un cop hem fet malloc només cal accedir a una posició més de les que té el vector. Per exemple, en el nostre cas, fem un malloc de 10,000 posicions, per tant si intentem accedir a la posició 10,001 del nostre vector, estarem fent un accés invàlid i executant el programa amb valgrind, aquest ens donarà un missatge d’error.

Exemple:

```
int *p_int = (int*) malloc(linies*sizeof(int));
printf("%d",p_int[10000]);
```

L’error que dóna valgrind és:

```
Invalid read of size 4
  at 0x400924: main (in /home/xavier/Documents/UB_xavi/SO2/Practica1/src/p1_1)
Address 0x520ef30 is 0 bytes after a block of size 40,000 alloc'd
  at 0x4C2DB8F: malloc (in /usr/lib/valgrind/vgpreload_memcheck-amd64-linux.so)
  by 0x4008B2: main (in /home/xavier/Documents/UB_xavi/SO2/Practica1/src/p1_1)
```