TD Mémoire virtuelle

Décembre 2019

Programme et mémoire

```c
#include <stdio.h>
#include <stdlib.h>

int a=42;

void f(int p){
    int b=3;
    int c=b+p;
    printf("c=%d\n",c);
    a=c;
    getchar();
}

int main(){
    int *d=(int*)malloc(sizeof(int));
    *d=7;
    f(a+*d);
    return 0;
}
```

1. Quelles zones mémoires sont utilisées par ce programme et pour quoi faire?
2. Est ce que les adresses manipulées sont virtuelles ou physiques?

Mémoire virtuelle par pagination

3. Quels sont les avantages du mécanisme de mémoire virtuelle?
4. Considérons une architecture 32 bits, de combien d'adresses virtuelles dispose-t-on? Et sur une architecture 64bits?

5. Comment est structurée la mémoire virtuelle (on considère une architecture 32bits)? Faire un dessin.
6. Que gère la table des pages?
7. Qui lit la table des pages?
8. Que doit faire le processeur pour lire un octet en mémoire à l'adresse virtuelle 0xA48051?
9. Quel dispositif est mis en oeuvre pour accélérer cette tâche?
10. Quelle est la taille de la table des pages?

Bits de contrôle

Linux utilise un certain nombre de bits de contrôle pour coder les propriétés sur les pages.

- PAGE PRESENT | Page is resident in memory and not swapped out
- PAGE PROTNONE | Page is resident, but not accessible
- PAGE RW | Set if the page may be written to
- PAGE USER | Set if the page is accessible from userspace
- PAGE DIRTY | Set if the page is written to
- PAGE ACCESSED | Set if the page is accessed

11. Sachant que les blocs des adresses physiques font 4ko(mémoire alignée), où peuvent être stockés ces bits de contrôle?
12. Il existe un traitant pour chacun d'entre eux, indiquer ce que chaque traitant va traiter?
13. Dans quels cas le contenu de la mémoire auquel on souhaite accéder n'est pas en mémoire physique?
14. Comment ce contenu sera-t-il mis en mémoire, et dans quel état va se trouver le processus associé?