Pagination

Décembre 2008

1 Mémoire virtuelle

La mémoire associée à un processus par `linux` sur une architecture X86 32bits est limitée à 3Go. Il s'agit d'une mémoire virtuelle qui dépasse souvent la quantité de mémoire effectivement présente sur le système.

```
00db6000-00db7000 r-xp 00db6000 00:00 0          [vdso]
08048000-08049000 r-xp 00000000 fd:00 565068     mem.exe
08049000-0804a000 rwxp 00000000 fd:00 565068     mem.exe
09be5000-09c06000 rwxp 09be5000 00:00 0

b7f2c000-b7f59000 rw-p b7f2c000 00:00 0
b7f65000-b7f66000 rw-p b7f65000 00:00 0
bfeb4000-bfec9000 rw-p bfeb4000 00:00 0          [stack]
```

- A quoi sert le 1Go manquant ?
- Quelles sont les protections des zones code, data, pile et tas?
- Pour quelles raisons ces zones sont ainsi protégées ?

2 Principe de pagination

Le principe de pagination permet de patagée la mémoire physique entre les processus. Les adresses mémoires émises par le processeur sont des adresses virtuelles, indiquant la position d'un mot dans la mémoire virtuelle.

`getpagesize - Obtenir la taille des pages memoire du systeme.`

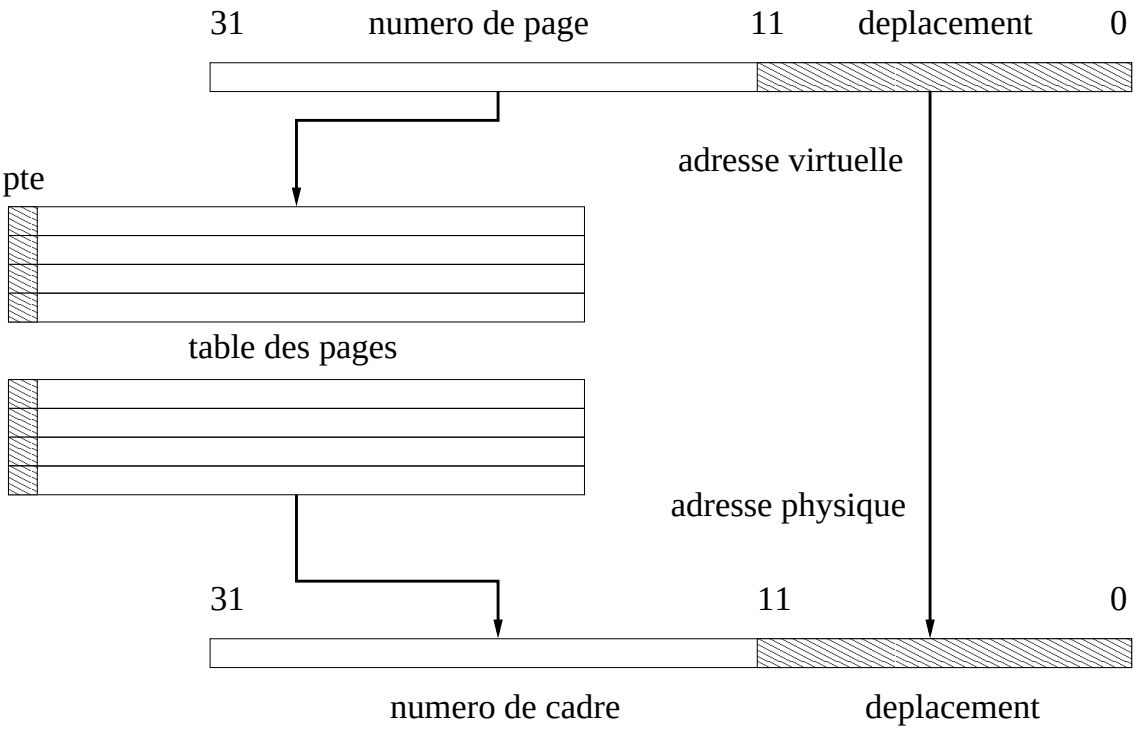
SYNOPSIS

```
#include <unistd.h>
size_t getpagesize(void);
```

DESCRIPTION La fonction `getpagesize()` renvoie le nombre d'octets dans une page.

HISTORIQUE

Cet appel est apparu en premier dans BSD 4.2.



- Rappeler les mécanismes de la pagination.
- Qu'est-ce qu'un défaut de page.
- Qu'est-ce que le principe de localié.
- Ensemble de travail.

3 Duplication

```
1
2  void procij(char **mat, int nl, int nc)
3  {
4  int i, j;
5  for(i = 0; i < nl; i++)
6    for(j = 0; j < nc; j++)
7   mat[i][j] = (i + j) % 256;
8  }
9  void procji(char **mat, int nl, int nc)
10 {
```

```
11  int i, j;
12  for( j = 0; j < nc; j++)
13    for( i = 0; i < nl; i++)
14   mat[ i ][ j ] = (i + j) % 256;
15  }
16
17  int main( int argc, char *argv[])
18  {
19    int nl, ps, cpt, mode;
20    int i;
21    char ** mat;
22
23     ps = getpagesize();
24     nl = atoi( argv[1]) * 10;
25    cpt =  atoi( argv[2]);
26    mode= atoi(argv[3]);
27    while ( cpt--) {
28      mat = ( char** ) calloc( nl, sizeof( char* ) );
29      for( i = 0; i < nl; i++ )
30        mat[ i ] = ( char*) calloc( ps , sizeof( char ) );
31
32      if ( mode ) procij( mat, nl, ps);
33      else procji( mat, nl, ps);
34      for( i = 0; i < nl; i++ )
35        free( mat[ i ] );
36      free( mat );
37      }
38
39    return 0;
40  }
```

4 Principe de pagination

Les adresses mémoires émises par le processeur sont des adresses virtuelles, indiquant la position d'un mot dans la mémoire virtuelle. Cette mémoire virtuelle est formée de zones de même taille, appelées pages. Une adresse virtuelle est donc un couple (numéro de page, déplacement dans la page). La taille des pages est une puissance de deux, de faon à déterminer sans calcul le déplacement (10 bits de poids faible de l'adresse virtuelle pour des pages de 1024 mots), et le numéro de page (les autres bits). * La mémoire physique est également composée de zones de même taille, appelées "cadres" (frames en anglais), dans lesquelles prennent place les pages. * Un mécanisme de traduction (translation, ou génération d'adresse) assure la conversion des adresses virtuelles en adresses physiques, en consultant une table des pages (page table en anglais) pour connaitre le numéro du cadre qui contient la page recherchée. L'adresse physique obtenue est le couple (numéro de cadre, déplacement). * Il peut y avoir plus de pages que de cadres (c'est là tout l'intérêt) : les pages qui ne sont pas en mémoire sont stockées sur un autre support (disque), elles seront ramenées dans

un cadre quand on en aura besoin.