Exo1 :

1. ps -eo pid,vsz,rss,%mem

2. free affiche la quantité totale de la mémoire physique libre, utilisée et de la partition swap,

free -s 5

3. head -n 1 /proc/meminfo

4. La commande vmstat fournit des informations à propos des processus, de la mémoire, de la pagination, des entrées-sorties, des interruptions et de la répartition du temps CPU.

5. cette commande donne un grand nombre d'informations sur l'état actuel du processus (entre autre : l'état de la mémoire du processus),

Exo 2

Résultat avec -static : (ça dépond de la machine, et autre facteurs !!!)

PID = 3425
adresse de une_globale = 80ed0a8
adresse de une_autre_globale = 80eb068
adresse de une_locale = bfb8502c
adresse de alloc = b6d39008
adresse de main = 8048e44
adresse de printf = 804f7b0
08048000-080ea000 r-xp 00000000 08:06 437029 /home/belfedhal/exom
080ea000-080ec000 rw-p 000a1000 08:06 437029 /home/belfedhal/exom
080ec000-080ee000 rw-p 00000000 00:00 0
08d96000-08db8000 rw-p 00000000 00:00 0 [heap]
b6337000-b773a000 rw-p 00000000 00:00 0
b773a000-b773c000 r--p 00000000 00:00 0 [vvar]
b773c000-b773e000 r-xp 00000000 00:00 0 [vdso]
bfb66000-bfb87000 rw-p 00000000 00:00 0 [stack]

1.

L'exécution de ce programme provoque l'affichage d'un PID et d'une série d'adresses. Dans le système Linux, les utilisateurs sont capables d'obtenir beaucoup d'informations sur les processus.
Ces informations se trouvent dans le répertoire /proc/PID_du_processus . On y trouve notamment le fichier maps qui donne la liste des régions associées à ce processus. Pour chaque région nous trouvons son espace adressable, les protections, l'offset (le

décalage), le numéro du périphérique (majeur:mineur) et un numéro d' i-node. On trouve également dans ce répertoire le fichier statm qui donne des statistiques sur l'utilisation de la mémoire (man proc pour avoir plus de précisions).

08048000-080ea000:programme principal (.text) + données en lecture seule
080ea000-080ec000: données initialisées (.data)
080ec000-080ee000 :données non-initialisées (.bss)
08d96000-08db8000: tas
bfb66000-bfb87000: pile

2.

PID = 7520
adresse de une_globale = 804a03c
adresse de une_autre_globale = 804a030
adresse de une_locale = bfbda2dc
adresse de alloc = b6bed008
adresse de main = 80484dd
adresse de printf = 8048370
08048000-08049000 r-xp 00000000 08:06 437061 /home/belfedhal/exom
08049000-0804a000 r--p 00000000 08:06 437061 /home/belfedhal/exom
0804a000-0804b000 rw-p 00001000 08:06 437061 /home/belfedhal/exom
b6bed000-b75ef000 rw-p 00000000 00:00 0
b75ef000-b7797000 r-xp 00000000 08:06 523361 /lib/i386-linux-gnu/libc-2.19.so
b7797000-b7799000 r--p 001a8000 08:06 523361 /lib/i386-linux-gnu/libc-2.19.so
b7799000-b779a000 rw-p 001aa000 08:06 523361 /lib/i386-linux-gnu/libc-2.19.so
b779a000-b779d000 rw-p 00000000 00:00 0
b77b0000-b77b3000 rw-p 00000000 00:00 0
b77b3000-b77b5000 r--p 00000000 00:00 0 [vvar]
b77b5000-b77b7000 r-xp 00000000 00:00 0 [vdso]
b77b7000-b77d7000 r-xp 00000000 08:06 523337 /lib/i386-linux-gnu/ld-2.19.so
b77d7000-b77d8000 r--p 0001f000 08:06 523337 /lib/i386-linux-gnu/ld-2.19.so
b77d8000-b77d9000 rw-p 00020000 08:06 523337 /lib/i386-linux-gnu/ld-2.19.so
bfbbc000-bfbdd000 rw-p 00000000 00:00 0 [stack]

Les régions mémoire :

08048000-08049000 :programme principal (.text) + données en lecture seule
0804a000-0804b000: données initialisées (.data) et données non-initialisées (.bss)
b6bed000-b75ef000: tas
b75ef000- b779a000: libraire dynamique standard du langage C
b77b7000-b77d9000: chargeur initial des librairies dynamiques
bfbbc000-bfbdd000: pile

3.

La carte mémoire n'a pas été changée car la nouvelle allocation de mémoire dynamique se trouve dans le « heap »

Exo3

1.

```
#include <stdlib.h>

char* tab;

int main()
{
tab = malloc(16384* sizeof(char));

for (int i=0; i<16384; i++) tab[i]='a';

return 0;
}
```

A compiler avec -std=c99 :
gcc -std=c99 -static exo3m.c -o exo3m

2.

```
#include <sys/types.h>
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>
#include <sys/mman.h>
#include <malloc.h>


char* tab;
char* alloc;
int pagesize;

int main()
{
alloc = malloc(1024L * 1024L );
tab = malloc(16384* sizeof(char));

for (int i=0; i<16384; i++) tab[i]='a';

pagesize = sysconf(_SC_PAGE_SIZE);
```

```
printf("début du tableau:  0x%lx\n", (long) tab);

tab = memalign(pagesize, 4 * pagesize);

mprotect(tab+ pagesize, 2 * pagesize, PROT_WRITE);  /* la mémoire peut être
accédée uniquement écriture*/
/*Pour la lecture :  PROT_READ*/

printf("pagesize: %d\n", pagesize);


printf ("%c\n", tab[4098]); /* Segmentation fault, car la mémoire est protégée contre la
lecture*/
/* pour tester la protection contre l'écriture, on peut mettre  tab[4098]='b';*/

sprintf(alloc, "cat /proc/%d/maps", getpid());
system(alloc);

return 0;
}
```

3.

Carte mémoire après la protection :

```
08048000-080ea000 r-xp 00000000 08:06 437065     /home/belfedhal/exo3m
080ea000-080ec000 rw-p 000a1000 08:06 437065     /home/belfedhal/exo3m
080ec000-080ee000 rw-p 00000000 00:00 0
08f9f000-08fa6000 rw-p 00000000 00:00 0          [heap]
08fa6000-08fa8000 ---p 00000000 00:00 0          [heap]
08fa8000-08fc1000 rw-p 00000000 00:00 0          [heap]
b7669000-b776b000 rw-p 00000000 00:00 0
b776b000-b776d000 r--p 00000000 00:00 0          [vvar]
b776d000-b776f000 r-xp 00000000 00:00 0          [vdso]
bfdbd000-bfdde000 rw-p 00000000 00:00 0          [stack]
```

Avec l'option PROT_NONE de la fonction mprotect, on a enlevé toutes les autorisation (lecture, écriture et exécution)