

TP Systèmes d'Exploitation : mémoire

Responsable

Philippe SWARTVAGHER

philippe.swartvagher@enseirb-matmeca.fr

Intervenante

Fadwa ABAKARIM

fadwa.abakarim@enseirb-matmeca.fr

2025 – 2026

Grandement inspiré du TP de Brice GOGLIN

1 Processus et mémoire

1. Observez le contenu du fichier `/proc/self/maps`. Que contient ce fichier ?
2. Lancez la commande suivante :

```
watch -n 1 cat /proc/self/maps
```

- Que fait cette commande ? Qu'observez-vous ? *Indice* : cherchez du côté de *ASLR*.
3. Écrivez un programme C qui fait des `malloc` en boucle tant que `malloc` ne renvoie pas `NULL` en multipliant à chaque fois par deux la quantité de mémoire que doit allouer `malloc`. À chaque itération de la boucle, le programme dormira deux secondes. Au début du programme, le programme affichera son PID et dormira plusieurs secondes. À la sortie de la boucle, le programme dormira à nouveau un moment.
 - (a) Lancez le programme et observez dans `htop` comment évoluent les valeurs liées à la mémoire du processus. Est-ce normal ?
 - (b) Relancez le programme en utilisant `strace`. Comment est en réalité allouée la mémoire de `mallocs` qui demandent beaucoup de mémoire ? Pourquoi pour des `mallocs` plus petits, il n'y a pas d'appel système lié à la mémoire ?
 4. Écrivez un programme qui fait un `malloc` pour représenter un tableau de `4096*4096` chars. Faites un `printf`, dormez quelques secondes. Ensuite, faites une boucle pour remplir les 4096 premières cases du tableau par une valeur constante. Une fois la boucle terminée, faites un `printf` et dormez quelques secondes.
 - (a) Exécutez le programme et observez les valeurs liées à la mémoire dans `htop`.
 - (b) Exécutez le programme avec `/usr/bin/time` et notez le nombre de défauts de page.

- (c) Changez la boucle du programme de façon à ne pas initialiser les 4096 premières cases du tableau, mais toutes les cases multiples 4096. Observez maintenant le comportement du programme avec `htop` et `/usr/bin/time`, comparez les valeurs obtenues avec la précédente version du programme et expliquez les différences.
- (d) En utilisant la fonction `gettimeofday` vue dans le TP précédent, mesurez la durée de la boucle dans les deux versions. Comparez et expliquez les durées obtenues.

2 Taille d'un fichier et occupation disque

Pour éviter d'écrire plein de programmes en C, on pourra utiliser l'outil `dd` pour remplir des bouts de fichier. Par exemple, pour copier 1 bloc (`count`) de taille 1 (`bs`) du fichier `/dev/zero` (`if`) vers le fichier `fichier` (`of`) en sautant 1000000 blocs (`seek`), on pourra par exemple utiliser :

```
dd if=/dev/zero of=fichier count=1 bs=1 seek=1000000
```

1. Créez un fichier contenant un seul octet. Observez sa taille avec les commandes `du -h` et `stat`. Expliquez les différents chiffres observés.
2. Agrandissez votre fichier jusqu'à ce que l'occupation disque augmente et expliquez ce qu'il se passe. Continuez plusieurs fois et essayez de prédire l'occupation disque en fonction de la taille réelle.
3. Obtenez des fichiers de même taille, mais en jouant sur le paramètre `bs`. Qu'observez-vous ?
4. Créez un fichier de taille nulle et expliquez comment son occupation disque peut être nulle.
5. Créez un fichier ne contenant qu'un octet à la position 10000. Expliquez sa taille et son occupation disque.

3 `mmap` et *copy-on-write*

Récupérez ce fichier.

1. Que fait le code présent dans ce fichier ? Exécutez ce code et justifiez le résultat obtenu.
2. Changez le second appel à `mmap` pour utiliser `MAP_PRIVATE` au lieu de `MAP_SHARED`. Exécutez le code ainsi obtenu et justifiez le résultat obtenu.
Indice : le titre de cet exercice.
3. Essayez différents ordres d'écriture dans les deux `mmaps` : observez-vous des différences ? Pourquoi ?

Essayez notamment les ordres suivants :

- 1111, 4444, 2222, 3333
- 1111, 2222, 4444, 3333