Document **Système** et réseau - Conception des systèmes Licence 3

Informations mémoire - Linux

V. Felea

Ce document résume les informations essentielles concernant la gestion mémoire sous Linux.

Pour information, la version courante de la distribution utilisée peut être obtenue par

- la commande `lsb_release -a`
- le fichier `/etc/issue`

La version du noyau : `uname -r` / fichier `/proc/version`.

Toutes les commandes/fichiers donnés dans ce document ont été testés sous la version du noyau Linux deployée dans les salles de TP (en septembre 2017 - 4.4.0-31-generic). Certaines commandes exigent les droits administrateur donc ne peuvent pas être testés sur les machines à l'université.

Remarque. D'une version à l'autre du noyau, ainsi que d'une distribution à l'autre, les fichiers peuvent changer, ainsi que les commandes accessibles par défaut.

1 Mémoire physique et swap

Commande `free` : la quantité de mémoire physique libre et occupée (intégrant le swap)

Swap (https://doc.ubuntu-fr.org/swap)

L'espace d'échange, aussi appelé par son terme anglais *swap space/swap*, est une zone d'un disque dur faisant partie de la mémoire *virtuelle* de l'ordinateur. Il est utilisé pour décharger la mémoire vive physique (RAM) de l'ordinateur lorsque celle-ci arrive à saturation. L'espace d'échange, dans Ubuntu, se trouve généralement sous une forme de partition de disque dur, appelée partition d'échange. Il peut aussi se présenter sous forme de fichier, appelé fichier d'échange (depuis Ubuntu 17.04 - fichier présent dans /swapfile).

Taille d'un espace d'échange

1. **Source Ubuntu (https://doc.ubuntu-fr.org/swap)**
   Votre ordinateur dispose de 1 Gio de RAM ou plus ? Allouez un espace d'échange de 1× à 1,5× la taille de votre RAM.
   Votre ordinateur dispose de moins de 1 Gio de RAM ? Allouez un espace d'échange de 1,5× à 2× la taille de votre RAM.

2. **Source RedHat**
(`https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/6/html/Installation_Guide/s2-diskpartrecommend-ppc.html#id4394007`)

Amount of RAM in the system	Recommended swap space	Recommended swap space (when hibernation)
2GB of RAM or less	2 times the amount of RAM	3 times the amount of RAM
2GB to 8GB of RAM	equal to the amount of RAM	2 times the amount of RAM
8GB to 64GB of RAM	0.5 times the amount of RAM	1.5 times the amount of RAM
64GB of RAM or more	4GB of swap space	no extra space needed

Commandes mémoire swap :

- `free` (même valeur que `/proc/swaps`)
- `swapon -s`
- `dmesg` (entrée `Memory`)

Fichiers :

- `/proc/meminfo` (entrée `SwapTotal`)
- `/etc/fstab` : montage swap
- `/proc/swaps` : vérifier que le swap existe

Remarque sur `/proc`. `/proc` est un pseudo-système de fichiers (système de fichiers virtuel). Aucune surface de stockage n'est rendue accessible via `/proc` mais seulement une hiérarchie d'objets implantés en mémoire auxquels sont associées des fonctions/méthodes.
Exemple :

```
> ls -l /proc/version
-r--r--r-- 1 root   root   0  sept.18 14:42  /proc/version
```

La commande `file /proc/version` donne un contenu vide.
La commande `cat /proc/version` donne le contenu attendu (car `cat` invoque `read`, routine du noyau, qui explore les éléments internes à Linux et retourne une chaîne de caractères au processus `cat`).

2 Mémoire virtuelle (par processus)

Commandes :

- `top` (champ `VIRT`) : mémoire virtuelle utilisée
- `ps -u` : 5ème colonne (`VSZ` = Virtual Memory Size) la taille (en Ko) de la mémoire virtuelle des processus
- `ulimit -a` : les limites associées à un processus (fixées dans le fichier `/etc/security/limits.conf`)
- `pmap [pid]` : image mémoire d'un processus (espace d'adressage)

  - option `-x`
  - identifier l'utilisation des segments de mémoire partagée
    `pmap -x [pid] | grep shmid` → descripteur du segment **en hexa**
  - mémoire ou bibliothèques partagées → Mode (s = shared)

Fichiers :

- `/proc/[pid]/stat` : informations sur l'état du processus
  Champs :
  - (24) RSS (Resident Set Size): taille (en nombre de pages) en mémoire pour le processus (incluant code, données et pile, excluant les pages non demandées ou déchargées sur le swap)
  - (36) nswap : nombre de pages déchargées sur le swap (not maintained)

- `/proc/[pid]/statm` : fournit des informations sur l'utilisation de la mémoire (**en nombre de pages**)

size	(1) taille totale de l'espace mémoire processus = `VmSize`/taille page (`VmSize` depuis `/proc/[pid]/status`)
resident	(2) = équivalent de RSS en kilo-octets = `VmRSS`/taille page (`VmRSS` depuis `/proc/[pid]/status`)
share	(3) pages partagées (i.e. information persistante par fichier (*))
text	(4) code
lib	(5) bibliothèques (non utilisée en Linux 2.6)
data	(6) données + pile
dt	(7) pages modifiées (dirty pages) (non utilisée en Linux 2.6)

  dirty pages = pages qui ont été modifiées depuis la dernière écriture sur disque

  (*)Le partage de la mémoire par fichier ou périphérique est le résultat de l'appel de la fonction `mmap` avec l'option `MAP_SHARED` (mécanisme POSIX de partage mémoire).

- `/proc/[pid]/status` : informations sur l'état du processus (mémoire incluse)

`VmSize` (kB)	état courant de l'utilisation de la mémoire virtuelle
`VmRSS` (kB)	`RSS`

  `VmSize-VmRSS` = mémoire non référencée au cours du programme

- `/proc/[pid]/maps` : cartographie mémoire du processus (la liste des régions associées à ce processus)
  - adresse début, adresse fin (espace d'adressage du processus)
  - droits d'accès (r/w/x), espace privé au processus (p) ou partagé (s)
  - déplacement (si la région a été mappée depuis un fichier, utilisant `mmap`, il s'agit du déplacement à l'intérieur du fichier où le mapping commence. Valeur à 0 si pas de mapping depuis un fichier.)
  - numéro de périphérique (si mapping depuis fichier)
  - numéro inode (si mapping depuis fichier)

– chemin d'accès (si mapping depuis fichier) ([heap], [stack], [vdso])
vdso = **v**irtual **d**ynamic **s**hared **o**bject (utilisé par des appels système pour passer en mode noyau)

Pourquoi l'adresse de début n'est pas 0x00000000 ?

L'éditeur de lien (éditeur statique de liens `ld`) assure que l'adresse de début du segment de code est toujours la même. Sa valeur dépend de l'architecture de la machine. Pour les processeurs x86, il s'agit de 0x08048000 (adressage 32-bits) et de 0x400000 (adressage 64-bits). Le segment de code est directement suivi par le segment de données. À l'exécution, la pile occupe la partie la plus haute de l'espace d'adressage (adresses les plus grandes) et évolue régressivement. Le tas et la zone des projections en mémoire des objets comme les segments de mémoire partagée, les segments pour l'éditeur dynamique des liens, les périphériques mappés en mémoire sont localisés dans l'espace d'adressage du processus entre le segment de données et la pile.

L'édition statique des liens est une technique qui permet qu'une application soit distribuée sans fichiers de bibliothèque (.dll, .so, .dylib). En fait, l'édition statique les inclut dans l'exécutable. On parle aussi de *compilation statique*.
Remarque. DLL sous Windows : Dynamic Link Library, SO sous Linux : Shared Object, DYLIB sous Mac OS : Dynamic Library

L'édition dynamique des liens lie l'application à une bibliothèque partagée chargée à l'exécution.

(readelf -l [obj.file])

```
#cat /proc/1/maps (droits administrateur)
address           perm offset  device inode    pathname    commentaire
08048000-0804e000 r-xp 00000000 03:01 64652    /sbin/init  text
0804e000-0804f000 rw-p 00006000 03:01 64652    /sbin/init  data
0804f000-08053000 rwxp 00000000 00:00 0                    zero-mapped BSS
40000000-40015000 r-xp 00000000 03:01 96278    /lib/ld-2.3.2.so  text
40015000-40016000 rw-p 00014000 03:01 96278    /lib/ld-2.3.2.so  data
40016000-40017000 rw-p 00000000 00:00 0                    BSS for ld.so
42000000-4212e000 r-xp 00000000 03:01 80290    /lib/tls/libc-2.3.2.so   text
4212e000-42131000 rw-p 0012e000 03:01 80290     /lib/tls/libc-2.3.2.so   data
42131000-42133000 rw-p 00000000 00:00 0                    BSS for libc
bffff000-c0000000 rwxp 00000000 00:00 0                    Stack segment
ffffe000-fffff000 ---p 00000000 00:00 0                    vsyscall page
```

La dernière colonne est un commentaire rajouté pour la compréhension de la sémantique de la région courante affichée.

Les droits administrateur ne sont pas requis pour l'accès au mapping mémoire des processus appartenant à l'utilisateur courant.

- `/proc/[pid]/smaps` : détail de maps

3 Pagination : table de pages / taille d'une page / défauts de page

Commandes :

- `getconf PAGESIZE` : taille d'une page (en octets)
- `ps -o min_flt,maj_flt pid`
  - `min_flt` = défaut de page, sans besoin de swap
  - `maj_flt` = défaut de page nécessitant des I/O
- `top` - option `f` (menu), sélectionner `nMin, nMaj`
- `/usr/bin/time` : exécute des programmes et résume l'utilisation des ressources
  option `-v` : les défauts de page
  Exemple : `/usr/bin/time -v ls /etc/resolv.conf`

  Commandes qui montrent le chargement de pages depuis le swap/disque :
  `/usr/bin/time -v xclock` (lancées deux fois de suite)
  (choisir toute commande peu susceptible d'avoir été utilisée auparavant)
- `vmstat` : statistiques sur le fonctionnement du système (dont la mémoire)
  - option `-s` : mémoire, CPU, changements de contexte, utilisation du swap pour les va-et-vient mémoire/disque
  - option `-m` : slabinfo (statistiques sur les données noyau allouées) (droits administrateur)
  - sans option : moyennes des statistiques depuis le dernier démarrage du système

  (+ info fichier `/proc/vmstat/` → `pgfault`, `pgmajfault`)
- `slabtop` / fichier `/proc/slabinfo`(droits administrateur) : utilisation de la mémoire par les structures du noyau (caches slab)

Fichiers:

- `/proc/[PID]/status` : entrée `VmPTE` = taille mémoire noyau utilisée par les tables de pages

4 Politique de remplacement de pages

Sous Linux, ce sont des variantes de LRU (Least Recently Used) : `https://www.kernel.org/doc/gorman/html/understand/understand013.html`

Plus particulièrement, il s'agit d'une combinaison de WSClock ([Tannenbaum,2002] `http://www.informit.com/articles/article.aspx?p=25260&seqNum=10`) et LRU-2Q ([T. Johnson, D. Shasha, 20th VLDB Conference 1994] `https://pdfs.semanticscholar.org/d62d/e5f995164fff50f5ce61c0113f6bc9f04225.pdf`).

Kernel Swap Daemon (`kswapd`) : thread noyau responsable de la sélection de pages à retirer de la mémoire centrale. Historiquement, il était activé toutes les 10 secondes, pour

décharger des pages de la mémoire sur le disque. Actuellement, il est activé par le gestionnaire d'allocation de pages quand le nombre de cadres libres passe en dessous d'un seuil (`page_low` (champ de la structure `zone_t` définie dans le fichier d'entête `linux/mmzone.h`). Ce seuil est le double d'un autre : `pages_low` = 2 × `pages_min`, ce dernier étant calculé au démarrage du système en fonction de la taille de la mémoire physique (=nbCadres/128).

Description du fonctionnement du thread de remplacement de page [Linux - Programmation système et réseau - Cours et exercices corrigés. Joëlle Delacroix. Édition : Dunod - 2012]

Le bit de référence et le champ d'âge d'une entrée de la table des pages sont utilisés par le thread `kswapd` pour choisir des pages victimes. (L'âge représente le temps passé en mémoire par la page.) Une page est victime si elle a atteint un âge donné (paramètre système) sans être référencée. Plus précisément :

- à chaque fois qu'une page est référencée, l'âge de la page devient égal à 0 et le bit de référence est mis à 1 ;
- à chacun de ses passages, le thread de remplacement de page met à 0 le bit de référence s'il est à 1 puis incrémente l'âge de la page.

Une page est victime si son bit de référence est à 0 et si elle a atteint l'âge limite. La page victime est écrite dans la zone de swap. Une adresse dans le périphérique de swap lui est alors affectée qui est composée du numéro du périphérique et de la position de la page dans la zone de swap. Cette adresse est sauvegardée dans l'entrée de la table des pages correspondant à la page. Ainsi, lors d'un défaut de page, le noyau utilise cette adresse pour trouver la page dans la zone de swap et la lire pour la ramener en mémoire centrale.

Fichiers :

- (répertoire `/proc/sys/vm/`) `swapiness`
  - `vm.swappiness` = 0 - Linux utilisera le disque dur (le swap) en dernière limite pour éviter un manque de mémoire vive
  - `vm.swappiness` = 60 - valeur par défaut de Linux : à partir de 40% d'occupation de RAM, le noyau peut copier une partie de la RAM dans la swap
  - `vm.swappiness` = 100 - tous les accès se font en écriture dans la SWAP (commande `sysctl` pour modifier le paramétrage)
- (répertoire `/proc/sys/vm`) `min_free_kbytes` : nombre minimum de koctets à garder libres à travers le système

D'une manière générale, le répertoire `/proc/sys/vm` contient des informations qui facilitent la configuration de la mémoire du noyau Linux.

5 Mémoire partagée (rappel et plus d'information)

Le fichier `/proc/sys/kernel/shmmax` contient la taille limite d'un segment de mémoire partagée qu'il est possible d'allouer sur le système (très utile dans le cas de plateformes multi-processeurs). Des segments de mémoire allant jusqu'à 4GB (sur une architecture 32bits, correspondant à

la valeur maximale d'un entier non signé), ou jusqu'à la valeur maximale d'un long non signé (sur une architecture 64bits) peuvent ainsi être créés.

Commande

- `df` (disk free) : afficher l'espace de mémoire libre/occupée (avec point de montage)

Système de fichiers en mémoire vive utilisé pour des données temporaires (pas de persistance) monté généralement sur `/dev/shm` (le chemin utilisée dans l'installation sur les postes des salles de TP est `/run/shm`).

`/dev/shm` : emplacement mémoire monté de la même manière qu'un disque dur, pour la gestion de la mémoire partagée POSIX

Commandes

- `lsof /dev/shm` (droits administrateur) : les processus qui utilisent la mémoire partagée (POSIX)
  Remarque cmd lsof. `lsof` donne des informations sur les fichiers, les processus et les connexions réseaux. Chaque ligne de la sortie correspond à un "fichier ouvert" (qui peut correspondre au répertoire courant du processus, à un fichier utilisé, à une bibliothèque partagée, à une connexion réseau, à un segment de mémoire partagée).
  Sortie :
  - `COMMAND` : le processus utilisant le fichier
  - `PID/TID` : identificateur de processus/thread
  - `USER` : utilisateur
  - `TYPE` : le type de noeud associé au fichier (GREG/GDIR/VDIR/VREG // IPV4 //DIR // FIFO // PIPE // etc)
  - `NAME` : nom de fichier ouvert

  `lsof` → considérer une des lib/*.so
  `lsof ....so` : affiche toutes les instances de programmes utilisant la bibliothèque `.so`
  `lsof -p pid` : affiche les fichiers ouverts par le processus d'identifiant `pid`

- `ipcs -m` : les segments de mémoire partagée System V
  `ipcs -m -l` : limites pour la mémoire partagée (System V)

Fichiers

- `/proc/sysvipc/shm` : informations sur les segments de mémoire partagée (SysV)
- `/proc/meminfo` - entrée `Shmem` : taille mémoire partagée (SysV, POSIX, tmpfs)

6 Fragmentation

Raison de la fragmentation interne liée à l'applicatif : *alignement mémoire* (*padding for alignment purpose*).

Considérons le cas d'une structure en mémoire. En théorie, les variables d'une structure sont placées les unes après les autres en mémoire, dans leur ordre de déclaration dans la

structure. Lorsque le processeur manipule un champ d'une structure, il commence par la lire depuis la mémoire en utilisant le bus de données. Celui-ci permet souvent de charger plusieurs octets depuis la mémoire. Les processeurs sont reliés à la mémoire vive par un bus de données plus large que la granularité de leur adressage pour augmenter leurs performances. Par exemple, un processeur de capacité d'adressage à un octet peut être relié à la mémoire vive par un bus de 4 octets. Alors, l'adressage d'une données de 4 octets doit se faire au multiple de 4. Il convient donc de bien aligner les données afin qu'il n'y ait pas de violation de contrainte (de multiplicité). Lorsque des données de taille différente sont enregistrées en mémoire les unes à la suite des autres, il peut être utile de laisser des trous entre elles afin qu'elles soient bien alignées. Il s'agit de l'alignement des données (data alignement/structure padding). C'est le cas de la pile et des types de données composées (comme les structures). La règle généralement appliquée est qu'une donnée doit se trouver à une adresse divisible par sa taille.

Exemple1

```
char* a = (char*) malloc(1);
char* b = (char*) malloc(1);
char* c = (char*) malloc(100);
char* d = (char*) malloc(100);
// printf les adresses (%f)
```

Output :

```
a = 0xdb64010
b = 0xdb64030  -> + 0x20
c = 0xdb64050  -> + 0x20
d = 0xdb640c0  -> + 0x70
```

Exemple2

```
struct Struct1 {    -> sizeof(struct Struct1) : 12
  char c1;          %fragm interne : 50%
  int x;
  char c2;
} ;

struct Struct2 {    -> sizeof(struct Struct1) : 8
  char c1,c2;       %fragm interne : 25%
  int x;
} ;
```

Fichier (droits administrateur)
`/sys/kernel/debug/extfrag/extfrag_index` : current fragmentation index

7 Segments de l'espace d'adressage du processus

Commande :
`size [a.out]` : affiche la taille des segments et la taille totale de chaque objet ;

- `text` - instructions machine que la CPU exécutera. En Linux, ce segment peut être partagé ;
- `data` - toutes les variables initialisées du programme (e.g. float salary=123.45;) ;
- `bss` - les données non initialisées comme les tableaux sans valeurs ou les pointeurs nuls.

Kernel Memory allocator : zone allocation / Buddy Allocation / SLAB Allocation - TO BE COMPLETED

Au démarrage du système, un premier allocateur appelé Boot Memory Allocator s'occupe d'initialiser les structures nécessaires à l'allocateur des pages physiques avant d'être remplacé par celui-ci.

Dans une architecture UMA (Uniform Memory Access) à chaque nœud sont associées trois zones mémoires physiques (qui peuvent être vides) (le cas d'une architecture 32b) :

- ZONE_DMA (pour l'accès DMA=Direct Memory Access des périphériques PCI) : les premiers 16Mo
- ZONE_NORMAL (directement mappée par l'espace d'adressage du processus) : entre 16mo et 896Mo
- ZONE_HIGHMEM (non mappée dans l'espace d'adressage virtuel

Fichiers : `/proc/pagetypeinfo`, `/proc/zoneinfo`

Le gestionnaire de mémoire du noyau Linux est construit sur deux niveaux. Le premier est un allocateur de pages physiques, appelé buddy. Celui-ci convient bien si les demandes mémoire s'expriment en multiples de pages. Il est par contre moins approprié s'il s'agit d'allouer quelques dizaines d'octets. D'où l'intérêt d'un deuxième allocateur, slab. Les sous-systèmes du noyau, les pilotes de périphériques, etc. passent par cet allocateur, qui gère des caches d'objets de taille spécifique (les descripteurs de processus, etc.), afin que leur allocation s'effectue rapidement et, cela, en minimisant la consommation mémoire.

Fichiers : `/proc/buddyinfo, /proc/slabinfo` (droits administrateur pour ce dernier fichier)
Commande : `slabtop` (droits administrateur)