Sujet de TP 6 **Système et réseau** - Gestion de processus et Numérotation IP Licence 3

Gestion de processus et adresses IP

V. Felea

Les exercices de ce sujet de TP couvrent, d'une part les notions d'ordonnancement de processus sous Linux, et d'autre part le mécanisme de base d'identification d'une machine en réseau, par l'adressage IP.

1 Système - processus

Généralités : Système de fichiers /proc

/proc est un pseudo-système de fichiers qui n'existe pas sur le disque. C'est le point de montage du pseudo système de fichiers du noyau. Il contient des fichiers permettant d'accéder aux informations sur le matériel, la configuration du noyau et sur les processus en cours d'exécution. La plupart des fichiers sont en lecture seule, mais quelques uns permettent la modification de variables du noyau. Le manuel de `proc`(5) peut être utile.

Les informations utiles du noyau :

Fichier	Description
`/proc/cpuinfo`	processeur (modèle, famille, taille du cache, etc.)
`/proc/meminfo`	mémoire physique (RAM), espace du swap, etc.
`/proc/mounts`	systèmes de fichiers montés
`/proc/devices`	périphériques disponibles
`/proc/modules`	les modules activés dans le noyau
`/proc/cmdline`	paramètres donnés au noyau lors de la mise en route
`/proc/version`	version du noyau

Les informations utiles du processus : `/proc/[pid]`.

Le répertoire `/proc/sys` contient des valeurs de différents paramètres du noyau, dont certaines peuvent être modifiées en temps réel. À la différence de tous les autres fichiers de `/proc`, certains fichiers de ce répertoire sont accessibles en écriture, mais seulement avec les droits administrateur.

- fichier `/proc/sys/kernel/threads-max` : nombre maximum de threads par processus pouvant être créés

Extrait du fichier source du noyau `linux/kernel/fork.c` :

```
/* The default maximum number of threads is set to a safe
 * value: the thread structures can take up at most half of memory. */
max_threads = mempages / (8 * THREAD_SIZE / PAGE_SIZE);
```

Le nombre maximum de threads varie donc selon les systèmes, en fonction de la taille de la mémoire vive.

- fichier `/proc/sys/kernel/pid_max` : la valeur à partir de laquelle la numérotation des PID reprendra à sa valeur initiale (ce qui signifie que la valeur dans ce fichier est celle du PID maximum plus un). Valeur par défaut pour les plateformes à 32 bits : 32768 ; pour les plateformes 64 bits, la limite est de 2^{22}.

La modification des paramètres système du noyau (si les droits le permettent) peut être faite aussi grâce à la commande `systcl`.
Remarque. L'option `-a` pour la commande `sysctl` permet de visualiser ceux listés sous `/proc/sys`.

Les variables de configuration du système peuvent être consultées grâce à la commande `getconf`. Par exemple, les constantes

- `CHILD_MAX`, qui donne le nombre maximum de processus fils pouvant être créés par un processus,
- `PTHREAD_THREADS_MAX`, qui donne le nombre maximum de threads pouvant être créés par un processus.

Remarque. La fonction système équivalente est `sysconf`.

Questions

Q1 Donner la configuration prévue dans votre système pour les processus (paramètres du noyau et variables de configuration).

1.1 Informations sur les processus du système

Questions

Q2 Que fait la commande `pstree` ? Expliquer l'affichage obtenu par cette commande, en précisant éventuellement, des options supplémentaires pour son exécution.
Indication. Expliquer notamment la présence de l'étoile. Confronter cet affichage avec celui de la commande `ps`.

Q3 Quelle est la différence entre les commandes `ps` et `top` ?

Q4 Comment utiliser la commande `ps` pour obtenir la liste des processus présents dans le système, et plus particulièrement, leur identifiant, leur état et la commande correspondante au processus ? Quels sont les états possibles d'un processus ?
Remarque. Les commandes `ps` et `top` utilisent le `/proc` pour récupérer leurs informations.

Q5 Combien de processus sont en exécution sur la machine ? Donner la ligne de commande permettant de l'obtenir.

Q6 Proposer une solution pour obtenir une information dynamique sur le système (comme par exemple le nombre de processus en exécution) de manière itérative.

1.2 Architecture matérielle

Les informations sur l'architecture matérielle de la machine (nombre de processeurs, nombre de cœurs, fréquence processeur, mémoire cache, etc.) peuvent être obtenues :

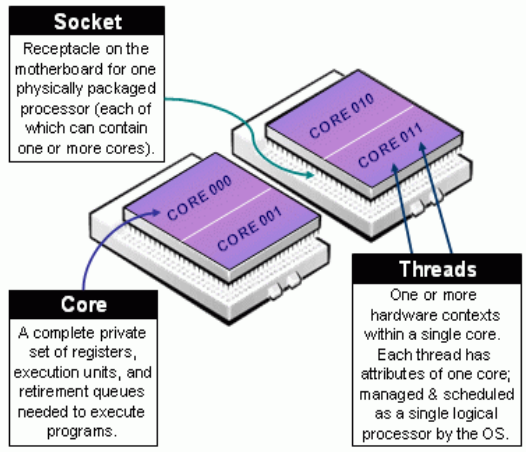
- par les commandes : `nproc`, `lscpu`, `lshw` (éventuellement des droits administrateur sont nécessaires)
- par le fichier `/proc/cpuinfo`

Succincte description architecturale

Le processeur ou CPU (Central Processing Unit) est le composant de l'ordinateur qui exécute les instructions. Le processeur peut être doté des plusieurs cœurs[1]. Un processeur standard possède un cœur (appelé *single-core*). Un processeur avec un seul cœur ne peut traiter qu'une seule instruction à la fois. Un processeur multi-cœur est composé de deux ou plusieurs cœurs indépendants, chacun étant capable de traiter des instructions individuellement.

Intel a créé une technologie appelée *hyperthreading* qui permet de créer deux unités de traitement logiques sur une seule puce : un cœur peut exécuter deux threads à la fois au lieu d'un seul. Par exemple, un Core i3 (bi-cœur) peut exécuter deux threads par cœur (au lieu d'un seul normalement) s'il possède l'hyperthreading, soit un total de quatre threads (au lieu de deux) ! Ainsi, même si les processeurs Core i5 sont quad-core, s'ils n'ont pas la technologie hyperthreading, le nombre de threads qu'ils peuvent traiter en même temps est à peu près égal à celui des Core i3 disposant de l'hyperthreading. Néanmoins, en pratique, quatre vrais cœurs sont beaucoup plus performants que deux cœurs + deux cœurs logiques (par hyperthreading). Le gain de la technologie hyperthreading est d'environ 30% pour un même processeur sans hyperthreading.

D'un point de vue physique, les cœurs d'un processeur peuvent être répartis sur des ensembles physiques différents, les puces (placées sur des *sockets* - voir l'illustration des notions dans la figure ci-contre). C'est notamment le cas de tous les processeurs multi-cœur basés sur la micro-architecture Intel Core2.



Le terme multiprocesseur indique que l'ordinateur possède sur sa carte mère plusieurs processeurs physiques. Dans un système multiprocesseur, chaque unité de calcul dispose de

[1] IBM commercialise le premier processeur multi-cœur en 2001, un POWER4.

son propre bus interne et de sa propre mémoire cache, ce qui n'est pas forcément le cas d'un processeur multi-cœur où les ressources peuvent être partagées. À fréquence égale, une machine à deux processeurs est aujourd'hui plus performante qu'un seul processeur double cœur.

L'intérêt du multi-cœur est le rapport performance / prix. Graver deux cœurs sur une même puce coûte moins cher que d'utiliser deux disques de silicium (wafers) pour réaliser deux processeurs distincts.

Il est également possible de combiner multi-cœur et multiprocesseur afin de gérer plus finement ses ressources matérielles (voir figure 1).

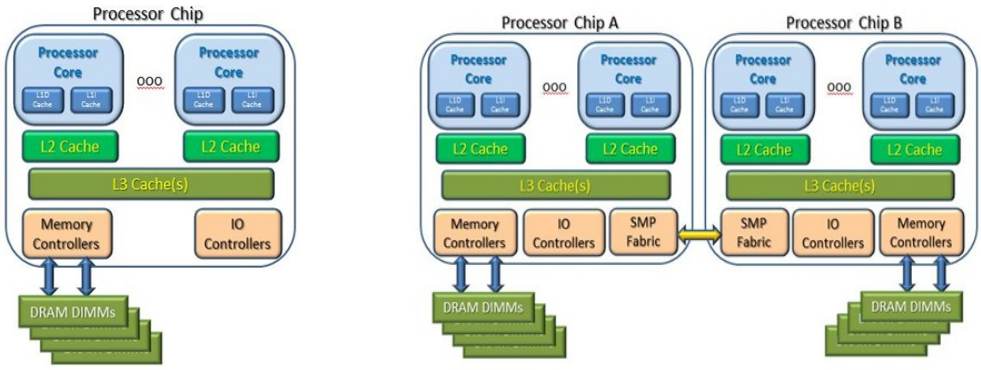


Figure 1: Architecture multi-cœurs/multi-processeurs

Questions

Q7 Donner les commandes système et leur résultat permettant d'obtenir les informations suivantes sur la machine courante utilisée : le nombre de supports (sockets/processeurs), le nombre de cœurs, si la technologie hyperthreading est implémentée. Plusieurs commandes différentes sont possibles. En donner deux.

Q8 Récupérer les caractéristiques du ou des processeurs de votre machine à l'aide du fichier `/proc/cpuinfo` par un programme en C. Identifier le nombre de processeurs physiques, de cœurs physiques et de cœurs logiques.

Remarque. Les deux questions précédentes concernent la même fonctionnalité, l'une obtenue par des commandes systèmes, l'autre par programmation en C. Étant redondantes, vous pouvez proposer une solution pour une seule d'entre elles.

1.3 Ordonnancement Linux

1.3.1 Description

Le noyau Linux ordonnance les processus selon 5 types de politiques d'ordonnancement :

- `SCHED_NORMAL/SCHED_OTHER` - pour les processus normaux (non temps réel) ;
- `SCHED_BATCH` - pour les processus de type batch (traitement par lots) qui sont des processus intensifs CPU, sans être interactifs. Ces processus ne seront pas interrompus aussi rapidement que les processus normaux, s'exécutant ainsi plus longtemps et profitant mieux des mémoires caches.

- `SCHED_IDLE` - pour certains processus système avec l'objectif de ne pas perturber l'exécution des autres processus ;
- `SCHED_FIFO` et `SCHED_RR` - pour les processus temps réel souple (soft real time).
  - `SCHED_FIFO` (politique principale d'ordonnancement des processus temps réel) - algorithme FCFS. Dans ce type d'ordonnancement, un processus en exécution continue à l'être jusqu'à ce qu'il libère le processeur volontairement, bloque ou est interrompu par un autre processus temps réel de priorité supérieure. Le quantum n'est pas utilisé. Les processus de plus faible priorité ne seront pas exécutés tant que la CPU n'est pas libérée. À priorité égale, les processus de cette classe ne se préemptent pas.
  - `SCHED_RR` est similaire à `SCHED_FIFO`, sauf que ces processus se voient allouer un quantum de temps en fonction de leur priorité.

Priorité À chaque processus est affectée une priorité, un entier, entre -20 et 19 (la valeur par défaut étant 0). Cette valeur est appelée *nice*. Plus la valeur est grande, moindre est la priorité. La commande `ps` (option `l`) permet de visualiser cette valeur (la colonne `NI`). La commande `renice` permet de modifier la priorité courante d'un processus. De même, la commande `nice` permet de lancer un programme avec une certaine priorité.

Dans le noyau, les priorités varient entre 0 et (`MAX_RT_PRIO`+39), où `MAX_RT_PRIO`=100. Plus précisément,

- les priorités des processus temps réel varient entre 0 et (`MAX_RT_PRIO`-1) inclus,
- les priorités des processus normaux varient entre `MAX_RT_PRIO` et (`MAX_RT_PRIO` + 39). Par défaut les valeurs nice entre -20 et 19 correspondent aux priorités noyau entre 100 et 140 (non inclus).

Les quanta sont ajustés en conséquence (dépendant de la priorité courante, niveau noyau, du processus).

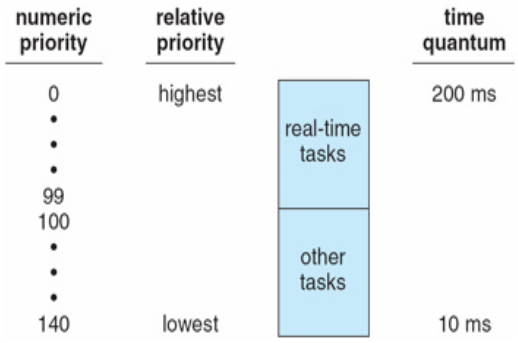


Figure 2: Priorités noyau en Linux et quanta

La priorité noyau d'un processus peut être obtenue grâce à la colonne `pri_baz`.
Note. Toutes les priorités sous Linux (introduites pour ajuster les priorités dans certains intervalles et pour la compatibilité POSIX et avec d'autres systèmes) :

- `priority` : -100..39
- `intpri/opri` (= 60 + `priority`) : -40..99
- `pri_foo/nice` (= `priority` - 20) : (valeur `nice`) -120..19
- `pri_bar` (= `priority` + 1) : -99..40
- `pri_baz` (= `priority` + 100) : (priorité noyau) 0..139
- `pri` (= 39 - `priority`) : 0..139
- `pri_api` (-1 - `priority`) (correspond aux priorités API Real-time) : -40..99

Les noms ci-dessus peuvent être utilisés comme noms de colonnes dans la configuration de la commande `ps`.
Remarque. Les commandes `ps` et `top` affichent des valeurs différentes de priorité (voir question 9 pour les identifier).

Algorithme d'ordonnancement Linux pour les processus normaux
À partir de la version Linux 2.6.23[2] (datant du 9 octobre 2007, auteur Ingo Molnár), c'est l'algorithme CFS [https://www.ibm.com/developerworks/library/l-completely-fair-scheduler/, http://people.redhat.com/mingo/cfs-scheduler/sched-design-CFS.txt] qui est utilisé par Linux pour l'ordonnancement des processus normaux (fichier `kernel/sched/fair.c` de la distribution, voir www.kernel.org).

1.3.2 Informations sur l'ordonnancement

Commandes :

- `chrt` : obtenir/modifier les priorités/la classe d'ordonnancement des processus en temps réel
- `nice`/`renice` : contrôler la priorité des processus de la classe d'ordonnancement `SCHED_NORMAL` / `SCHED_OTHER`
- `taskset` : récupérer ou fixer l'*affinité CPU* d'un processus en cours d'exécution en donnant son identificateur ou lancer une nouvelle commande avec une affinité CPU fournie. L'affinité CPU est une propriété de l'ordonnanceur qui attache un processus à un ensemble de CPUs d'une architecture matérielle.

Fichiers :

- `/proc/schedstat` : statistiques de l'ordonnanceur Linux pour l'ensemble du système
- `/proc/[pid]/schedstat` : statistiques de l'ordonnanceur Linux par processus

  à consulter https://www.kernel.org/doc/Documentation/scheduler/sched-stats.txt
- `/proc/sched_debug` : valeurs des variables qui configurent le comportement de l'ordonnanceur, des statistiques CFS, et informations sur les files (CFS, RT) sur les différents processeurs

[2] Actuellement, la version stable du noyau Linux est 4.13

D'autres statistiques, sur la charge de la machine, sont données par :

- le fichier `/proc/loadavg` : les premières trois valeurs donne la charge du système pour les périodes suivantes : dernière minute, dernières 5 minutes, dernières 10 minutes. La quatrième colonne est le nombre de processus couramment en exécution et le nombre total de processus. La dernière colonne est l'identifiant du dernier lancé.
- le fichier `/proc/uptime` / la commande `uptime` : le temps depuis lequel une machine, ou un logiciel informatique, tourne sans interruption ; l'information sur la charge actuelle du système est également fournie.

  Remarque. Sous Linux, la charge moyenne est le nombre de processus en train d'utiliser le(s) processeur(s) ou en train d'attendre de pouvoir les utiliser.

  Interprétation de la charge. Comme administrateur des systèmes, une charge moyenne élevée doit être considérée comme inquiétante. Quand elle dépasse le nombre de cœurs, cela dénote une forte demande de cycles CPU, et quand elle est en-dessous du nombre de cœurs, alors cela traduit des CPUs sous-utilisées.

Questions

Q9 Grâce aux commandes système `ps` et `top` afficher la priorité des processus. Quel constat peut être fait concernant les valeurs de priorité affichées ?

Q10 Écrire un programme sollicitant pour un temps important des cycles CPU (processus CPU-bound). Un exemple type de ce type de traitement est une boucle (éventuellement imbriquée, ou doublement imbriquée) sur des compteurs qui itère l'instruction vide.

- Lancer ce programme. Quelle est sa priorité par défaut ?
- Changer sa priorité par une commande système. Quel est l'impact de cette modification ?
- En déduire un scénario différent pour mettre en évidence le changement de priorité.

1.3.3 Fonctions système liées à l'ordonnancement

Le noyau dispose d'un petit nombre d'appels système lui permettant de gérer le placement/l'attachement des processus (bibliothèque `sched.h`) :

- `sched_setaffinity`/`sched_getaffinity`
- `sched_getcpu` (glibc $>$ 2.6)
- `sched_getparam`
- `sched_rr_get_interval`
- `sched_getscheduler`/`sched_setscheduler`
- `nice` (bibliothèque `unistd.h`)

Questions

Q11 Ecrire un programme qui affiche le nombre de CPU utilisables (nombre de processeurs × nombre de cœurs).
Indication. À utiliser la fonction C `sysconf` avec, comme paramètre, la constante `_SC_NPROCESSORS_ONLN`.

Q12 Compléter le programme de l'exercice précédent afin de montrer le changement d'affectation d'un processus. Pour cela, le programme est une simple boucle infinie qui vérifie à chaque itération si le processus est toujours sur la même CPU.

Q13 Écrire un programme qui s'exécute forcément sur une CPU valide donnée.

2 Réseau - adresse IP

Ces exercices réseau sont orientés vers la manipulation de base d'un poste en réseau, quant à son adresse IP. Restreints en manipulation à cause des droits administrateur des commandes de configuration, ils ont l'objectif de visualiser uniquement la situation actuelle sur votre machine. Certaines questions seront à répondre sans manipulation, pour des raisons de droits restreints.

Q0 Comment identifier les éléments matériels des interfaces réseau du poste ?
Indication. Utiliser les messages système de démarrage (par la commande `dmesg`) qui recensent, entre autres, les informations relatives au chargement des modules de pilotage des interfaces réseau. La commande `lshw` permet également d'obtenir ce type d'information.

Q1 Donner les adresses MAC des périphériques Ethernet disponibles sur votre machine.

Q2 Donner la commande pour afficher la correspondance IP / adresse MAC des ordinateurs et périphériques connectés. Quel est le résultat de son exécution sur votre machine ?

Q3 Quelle est l'adresse IP de votre machine ? Quelle est la commande permettant de l'obtenir ? Répondre à la fois pour un système Linux que pour un système Windows.
Remarque. Sous Linux, le fichier `/etc/network/interfaces` contient la configuration des interfaces réseau.
Dans une configuration statique et manuelle, le contenu de ce fichier est généralement le suivant :

```
# loopback interface
auto lo
iface lo inet loopback
# primary network interface
auto eth0
iface eth0 inet static
address [adresseIP]
netmask [masque]
gateway [adressePasserelle]
```

Le mot clé "auto" indique que la carte `eth0` sera configurée au démarrage du système. L'activation de cette configuration est faite grâce à la commande : `service networking restart` (nécessitant des droits administrateur).

Il existe également des assistants graphiques pour la configuration des interfaces réseaux, comme `Network-Manager`. Avec son utilisation, le fichier `/etc/network/interfaces` ne doit contenir que la configuration de l'interface locale (`lo`).

Q4 Quel est le masque de votre réseau filaire ? Combien d'adresses IP peuvent être définies pour les machines de ce réseau ? Quelle est leur plage ?

Q5 Quelle est l'adresse de broadcast de votre réseau ?

Q6 La commande `ping` envoie un paquet à une machine ayant une adresse donnée. C'est une commande qui permet de tester si la connexion vers l'adresse IP en question est effective. Tester la commande `ping` avec l'adresse IP :

- 127.0.0.1,
- avec votre adresse,
- du poste d'un de vos voisins (si ce poste est accessible dans votre environnement),
- d'un PC de la salle qui est booté sous Windows.

Comment expliquez-vous vos résultats ?

Q7 Quelle serait la ligne de commande permettant de configurer l'interface Ethernet `eth1` avec :

- l'adresse IP 192.168.10.20 et
- le masque de sous-réseau 255.255.0.0