TP2: Commandes système, gestion des processus, gestion de la mémoire

Module
1A, 2A ou 3A

Vous rédigerez un rapport de TP que vous rendrez en fin de séance à votre encadrant. Dans ce rapport, vous indiquerez à chaque fois les expériences réalisées, justifierez le choix de vos paramètres et donnerez une part importante à l'analyse des résultats obtenus.

1 Informations générales sur le système

De nombreuses commandes permettent d'obtenir sous linux des informations sur le système.

1.1 Commandes `uname` et `uptime`

▷ **Question 1.** Noter les informations fournies par `uname` ainsi que leur signification.

▷ **Question 2.** Même question pour la commande `uptime`.

1.2 Système de fichiers /proc

/proc est un pseudo-système de fichiers qui n'existe pas sur le disque et il ne semble exister qu'au moment où on référence un fichier ou un sous-répertoire. /proc est utilisé comme interface avec les structures de données du noyau. La plupart des fichiers sont en lecture seule, mais quelques uns permettent la modification de variables du noyau. On peut utiliser les commandes habituelles comme `ls` pour connaitre le contenu d'un sous-répertoire du /proc, `cat` pour lire des informations à partir de fichiers. Pour la suite un `man proc(5)` vous sera utile.

▷ **Question 3.** Exécuter les commandes suivantes et donner une interprétation aux résultats :

- `cat /proc/sys/kernel/hostname`
- `grep cpu /proc/stat`
- `more /proc/loadavg`
- `cat /proc/uptime`
- `cat /proc/version`
- `cat /proc/sys/fs/file-max`
- `cat /proc/sys/fs/file-nr`

2 Gestion des processus

Dans cette section, nous allons nous intéresser plus spécifiquement à la gestion des processus.

2.1 Informations sur les processus

▷ **Question 4.** Quelle est la différence entre les commandes `ps` et `top` ?

▷ **Question 5.** Que fait la commande `pstree` ?

▷ **Question 6.** Comment utiliser la commande `ps` pour obtenir la liste des processus en première colonne et leur état en 2ème colonne ? Quels sont les états possibles ?

Remarque : les commandes `ps` et `top` utilisent le `/proc` pour récupérer leurs informations.

2.2 Processus et système de fichiers /proc

Des informations sur les processus en cours d'exécution peuvent être trouvées à partir du /proc. Un processus est représenté avec un sous répertoire de /proc avec comme nom son pid. En tapant `ls /proc`, un certain nombre de numéros s'affichent. Il s'agit des pid des processus du système.

▷ **Question 7.** Quelles sont les informations concernant un processus donné fournies par la commande `ls -l /proc/pid/` ? En consultant les pages du manuel, déterminer la signification de chacune des informations suivantes : `cmdline`, `cwd`, `environ`, `exe`, `fd`, `mem`, `stat`.

▷ **Question 8.** Comment obtenir l'ID du processus parent d'un processus donné ?

2.3 Priorités des processus

Sous linux, les priorités s'échelonnent de -20 à 19, mais seul le super-utilisateur peut spécifier des priorités négatives. Plus le nombre est grand, plus la priorité est faible.

▷ **Question 9.** Comment peut-on obtenir à l'aide de `ps` des informations sur la priorité des processus en cours ? Ces informations sur la priorité peuvent être retrouvées aussi avec `top` mais en temps réel.

▷ **Question 10.** Retrouver les champs PRI (priorité) et NI (gentillesse/courtoisie) dans le procfs.

Pour changer la priorité d'un processus, on dispose de la commande `nice` pour modifier la priorité d'un processus à son exécution.

▷ **Question 11.** Taper la commande : `nice +5 ps -l`. Que remarquez-vous au niveau de la colonne NI ?

▷ **Question 12.** Lancer les commandes suivantes. Quelle est la valeur affichée dans la colonne NI ? Conclure.

```
nice sleep 240 &
ps -l
```

▷ **Question 13.** La commande `renice` permet de changer la priorité d'un processus au cours de son exécution. Donner un exemple sur le modèle de la question précédente, pour montrer l'utilisation de `renice`.

3 Gestion de la mémoire

La dernière section porte sur la gestion de la mémoire, l'impact de l'organisation des structures et le phénomène d'écroulement.

3.1 Informations sur la mémoire

▷ **Question 14.** En utilisant `ps`, afficher la liste de tous les processus avec le format `commande tailleV tailleR Pmem` où `tailleV` est la taille de la mémoire virtuelle du processus, `tailleR` est la taille de la mémoire physique utilisée par le processus, et `Rmem` est le pourcentage de mémoire physique utilisée par le processus par rapport à la mémoire physique totale du système.

▷ **Question 15.** Que fait la commande `free` ? Comment faire pour obtenir un nouvel affichage toutes les 5 secondes ?

▷ **Question 16.** Le programme `free` utilise ***/proc/meminfo***. Comment utiliser ce dernier pour afficher sur une seule ligne la quantité de mémoire disponible sur le système ?

▷ **Question 17.** Quelles informations peut-t-on obtenir en utilisant la commande `vmstat` ? Noter en particulier celles relatives à la gestion de la mémoire.

3.2 Alignement des structures (de Michel Simatic)

Le but de cet exercice est d'étudier l'impact de l'organisation des structures de données sur les performances de la mémoire en termes de nombre de défauts de pages.

▷ **Question 18.** Récupérer ***/home/depot/2A/RSA-Maimour/align.tgz*** et compiler les sources à l'aide de la commande `make`.

▷ **Question 19.** Analyser le contenu de `useGetrusage.c` et déterminer son rôle.

▷ **Question 20.** Comparer la structure tableElement des fichiers align.c et nonAlign.c.

▷ **Question 21.** Exécuter align et nonAlign. Que remarquez vous par rapport à la taille d'un élément du tableau dans les 2 programmes ? Cela est dû à quoi d'après vous ?

▷ **Question 22.** Sachant qu'une page mémoire est de 4Ko, combien de pages mémoire sont nécessaires pour stocker le tableau dans les 2 programmes ? Retrouver ces valeurs par un calcul simple !

▷ **Question 23.** Donner une interprétation aux résultats obtenus.

3.3 Phénomène d'écroulement (de Michel Simatic)

▷ **Question 24.** Récupérer ***/home/depot/2A/RSA-Maimour/ecroul.tgz*** et compiler les sources à l'aide de `make`.

▷ **Question 25.** gentil.c, gentil2.c et mechant.c initialisent un tableau de N*4096 caractères. Comparer leur manière de travailler.

▷ **Question 26.** Déterminer à l'aide de la commande free, la quantité de mémoire physique disponible et la taille de l'espace disque disponible pour le swap

▷ **Question 27.** Dans constante.h, mettre NBMEG à 90% de la taille mémoire disponible et recompiler. Dans une fenêtre 1, exécuter la commande vmstat 1 et dans une fenêtre 2, taper :

- `/usr/bin/time gentil`
- `/usr/bin/time gentil2`
- `/usr/bin/time mechant`

▷ **Question 28.** Commenter l'évolution des champs r, b, free, si, so, id (dans la fenêtre 1) et les différences de temps d'exécution entre les différents processus. Au vue de la valeur de NBMEG, à combien de défauts de pages mineurs pouvait-on s'attendre ? Comparer avec le résultat affiché par time.

▷ **Question 29.** Même manip (n'oubliez pas le make !) et même question avec 75% de la mémoire physique totale pour NBMEG.

3.4 Algorithmes d'allocation mémoire (pour ceux qui avancent vite)

▷ **Question 30.** Proposer une implantation de l'algorithme des frères siamois (vu en TD). Le programme sera lancé avec en argument la taille totale de la mémoire. L'utilisateur pourra ensuite saisir la création ou la libération d'un bloc mémoire en précisant le nom du bloc (avec sa taille lors d'une création). Le programme affichera en retour l'état d'occupation de la mémoire.

▷ **Question 31.** Même question pour la version généralisée de l'algorithme (avec une taille des blocs suivant un système de Fibonacci : 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 987, ...).