

System

- Numero: 1
- Prof: Laskar Gabriel
- Date: 3 Octobre 2017

OS

- Mettre a disposition les ressources pour les autres ressources
- Permettre de manipuler une abstraction (plus simple)
- Un programme avec plus de privilege

Kernel

- C'est un bout de code privilegie qui s'occupe du management des ressources
- C'est lui qui pose les abstractions
- Il donne un point d'entree sur les ressources
- Il restreind l'accès au ressources pour le redistribuer aux autres
- Ressources:
 - Memoire
 - Processeur
 - Materiel (peripherique)

Processeur (CPU)

- ISA (Instruction Set Architecture)
- Il manipule des registres (Zones memoire tres limite ~64bits et non limite)
 - Il permet de faire du calcul
 - Il y a des registres generaux
 - Il y a des registres de Controle (Accessible uniquement au mode *Superviseur*)
- Bus d'adresse / Bus de donnees (une serie de fils)
- Les bus sont relies a un Controleur Memoire (CM). Il dispatche la memoire entre les devices (dont la RAM)
- Il y a deux modes d'execution
 - Un mode *Superviseur* (Il peut executer des instructions privilegies). Il peut donc acceder aux registres de controles
 - Un mode *User* (Il ne peut pas)

Pour minimiser les maj du kernel, l'un des principes de base est: * Essayer de separer le *mecanisme* de la *politique d'utilisation* (comment on s'en sert) * Mecanisme est cote kernel * La politique d'utilisation en Userland

Syscalls

- Ce sont les APIs du Kernel
 - API: Application Programming Interface, Comment on se sert d'un service. "C'est un protocole haut niveau"
 - On passe du mode user au system
 - Ils permettent l'abstraction

Demarage

- Utilise le ROM, composant contenant de code pour charger le kernel (qui est sur le disk de base)
- PC: Program Counter
- IP: Instruction Pointer
- Firmware: Bout de code present dans tous les composants
 - Il charge le kernel
- Bootloader
- Le kernel a une command line
- mount rootfs (mount est un syscall: cf mount(2))
 - rootfs (Root File System) C'est le /.
 - Ca peut etre par exemple /dev/sda1.
 - Puis on peut mount le home /home
- On appelle un programme `/sbin/init` Il va etre en chage de finir l'initialisation
 - Il y a plein de programme *init* different. Il est different en fonction de l'os de la distrib etc... (Il est cote userland).
 - Entre autre il initialise *stdin stdout*... Mettre en place le reseau...
 - C'est ocmplique du coup il va separer les taches (runlevel: cf man runlevel)
 - Il doit gerer la durer de vie des services
 - Il va lancer plein de daemon (services)
 - * Gerer l'heure
 - * Gerer le network

- * Gerer les logins

ISO

- C'est un File System. Read only
- Le format est dependant du system
- L'utilisation de *read* est une mauvaise idee car ca implique de lseek enormement.
- On pourrait utiliser *preadv*. Car il prend un offset.
- On prefere utiliser *mmap*. Va charger le fichier en memoire de maniere intelligente.

Exemple de mmap

- Utilise xxd pour lire un fichier binaire
- Utiliser strace pour verifier que l'on fait pas de la merde
- TIPS vim: Utilise **K** sur un mot cle pour ouvrir le man direct.

```
1 #include <stdlib.h>
2 #include <stdio.h>
3 #include <unistd.h>
4 #include <fcntl.h>
5 #include <sys/stat.h>
6 #include <err.h>
7
8 int main(int argc, char **argv)
9 {
10     int fd = open(argv[1], O_RDONLY);
11     if (fd < 0)
12         err(1, "fail");
13     struct stat stat;
14     if (fstat(fd, &stat) < 0)
15         err(1, "fail");
16
17     void *ptr = mmap(NULL, stat.st_size, PROT_READ, MAP_PRIVATE, fd, 0);
18     if (ptr == MAP_FAILED)
19         err(1, "fail");
20
21     // Plus besoin de fd
22     if (close(fd) < 0)
23         err(1, "fail");
24 }
```

```
25 // play with the map (For my readiso you have ot use given structures
   )
26 char *identifier = ptr + 0x8000 + 1;
27 printf("%.5s\n", 5, identifier);
28
29 // free
30 if (munmap(ptr, stat.st_size) < 0)
31     err (1, "fail");
32
33 return 0;
34 }
```

Example struct shell

```
1 static struct
2 {
3     const char *cmd;
4     int (*func)();
5     const char *usage;
6 } shell_cmds[] = {
7     { "help", shell_help, "display help"},
8     { "info", shell_info, "display info"}
9 }
```