EPU - Informatique ROB4
Informatique Système
Divers - stat, minils

Sovannara Hak, Jean-Baptiste Mouret
hak@isir.upmc.fr

Université Pierre et Marie Curie
Institut des Systèmes Intelligents et de Robotique (CNRS UMR 7222)

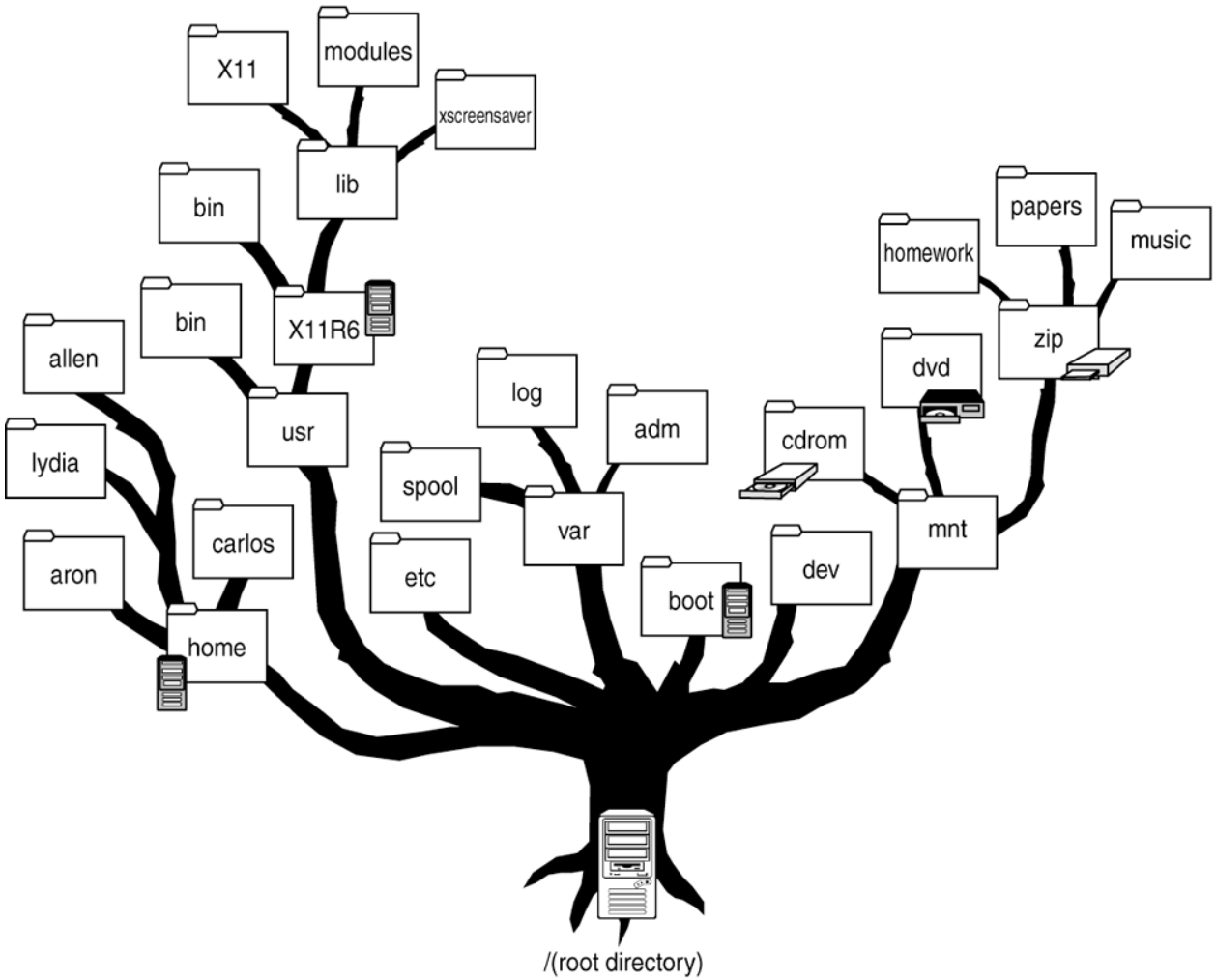
2014-2015

Plan de ce cours

1. Système de fichiers
   - Quelques rappels
   - Commandes du shell liées aux fichiers et aux répertoires
2. stat()
   - Utilisation de la fonction fstat() et stat()
3. Répertoires
   - Exploration d'un répertoire
   - Exercice : mini-ls

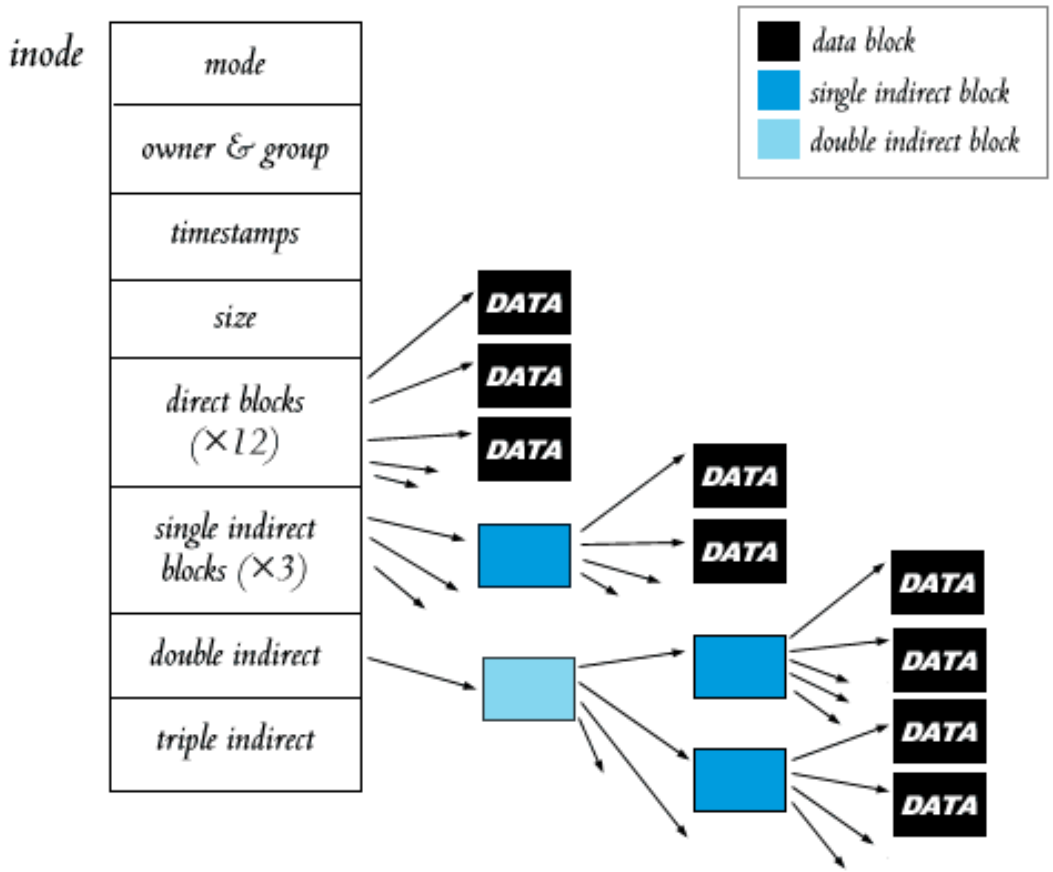
Rappel du plan

Système de fichier & OS



Exemple : i-nodes unix-bsd

Un i-node (index node) est une structure de donnée qui va représenter un objet du système de fichier (par exemple un fichier ou un répertoire). Elle contient notamment des metadonnées et des attributs.



Rappel du plan

`ls`

`ls` - liste le contenu d'un répertoire

- `ls -a` : ne pas masquer les fichiers commençant par .
- `ls -i` : *i* pour *inode*, affiche le numéro d'index
- `ls -l` : affichage long, pour avoir les permissions, le type, le propriétaire...
- `ls -R` : liste les sous-répertoires récursivement

La première colonne de `ls -l` donne le type et les droits du fichiers :

- premier caractère : type
  - - : fichier ordinaire
  - d : répertoire
  - l : lien symbolique
  - ...
- trois suivant : droits User.
  - r ou - : read
  - w ou - : write
  - x ou - : execute
- trois suivant : droits Group
- trois suivant : droits Other

chmod

- La commande shell permettant de modifier les droits d'accès à un fichier est `chmod`
- Elle permet de spécifier :
  - les droits de l'utilisateur propriétaire du fichier (u)
  - les droits des utilisateurs membres du même groupe (g)
  - les droits des autres utilisateurs (o)
- Elle permet d'autoriser ou d'interdire la lecture, l'écriture et l'exécution d'un fichier pour chaque type d'utilisateur. Elle permet aussi de donner les droits `Set - UID`.

- chmod MODE fichier
  - où MODE est de la forme `[ugoa]*([-+=]([rwxXst]*|[ugo]))+`
  - chmod uo+x fichier : ajouter droit x pour user et other
  - chmod a-w fichier : retirer droit w pour user, group et other

  - chmod OCTAL-MODE fichier
  - où OCTAL-MODE est un masque exprimé en octal
  - chmod 754 fichier : rwx r-x r–

`tree`

`tree` - liste le contenu des répertoires sous forme d'arbre

- `tree -a` : affiche tout les fichiers
- `tree -p` : affiche le type et les droits (comme ls -l)
- ... plus d'info sur `man tree` !

Rappel commande man

- q : quitter
- j, k : défilement ligne haut, bas
- d, u : défilement page haut, bas
- g, G : aller au début, fin
- /regex, ?regex : recherche regex
- n, N : résultat de recherche suivant, précédent

Ces raccourcis sont aussi valable pour `less` (et un peu pour `vim`)

Rappel du plan

Utilisation de la fonction fstat() et stat()

- Les attributs associés aux fichiers peuvent être récupérés par un appel aux primitives `stat()` et `fstat()` :
  `int stat(const char *ref, struct stat *infos);`
  `int fstat(const int desc, struct stat *infos);`
- Leur utilisation nécessite les inclusions suivantes :
  `#include <sys/types.h>`
  `#include <sys/stat.h>`
- Ces deux fonctions retournent les attributs associés à un fichier soit désigné par son nom (`stat()`) soit désigné par son descripteur (`fstat()`).
- Le type `struct stat` est défini dans `#include <sys/stat.h>` :

```
struct stat {
    dev_t     st_dev;     /* ID of device containing file */
    ino_t     st_ino;     /* inode number */
    mode_t    st_mode;    /* protection */
    nlink_t   st_nlink;   /* number of hard links */
    uid_t     st_uid;     /* user ID of owner */
    gid_t     st_gid;     /* group ID of owner */
    dev_t     st_rdev;    /* device ID (if special file) */
    off_t     st_size;    /* total size, in bytes */
    blksize_t st_blksize; /* blocksize for file system I/O */
    blkcnt_t  st_blocks;  /* number of 512B blocks allocated */
    time_t    st_atime;   /* time of last access */
    time_t    st_mtime;   /* time of last modification */
    time_t    st_ctime;   /* time of last status change */
};
```

Utilisation de la fonction fstat() et stat()

Le type de fichier codé dans le champ mode_t peut être obtenu en utilisant l'une des macros suivantes :

- S_ISBLK(infos->st_mode) , renvoie vrai si le fichier est un fichier spécial en mode bloc ;
- S_ISCHR(infos->st_mode) , renvoie vrai si le fichier est un fichier spécial en mode caractères ;
- S_ISDIR(infos->st_mode) , renvoie vrai si le fichier est un répertoire ;
- S_ISFIFO(infos->st_mode) , renvoie vrai si le fichier est un tube nommé ;
- S_ISREG(infos->st_mode) , renvoie vrai si le fichier est un fichier régulier ;
- S_ISLNK(infos->st_mode) , renvoie vrai si le fichier est lien symbolique ;
- S_ISSOCK(infos->st_mode) , renvoie vrai si le fichier est un socket ;

À propos de stat et fstat

- pas de permission requise sur le fichier en question
- droit x necéssaire à tout les répertoires de l'argument menant au fichier

```
#include <sys/types.h>
#include <sys/stat.h>
#include <unistd.h>

int stat(const char *path, struct stat *buf);
int fstat(int fd, struct stat *buf);
int lstat(const char *path, struct stat *buf);
```

- `stat()` : fichier désigné par path
- `fstat()` : fichier spécifié par le file descriptor
- `lstat()` : si path désigne un lien symbolique, les informations seront extraites pour le lien et non le fichier pointé par le lien

Exemple

```
#include <sys/types.h>
#include <sys/stat.h>
#include <stdio.h>
#include <time.h>

main() {
  struct stat info;

  if (stat("/home/shak/un_truc.txt", &info) != 0)
    perror("stat() error");
  else {
    puts("\nstat(): information about /home/shak/un_truc.txt:");
    printf("  inode:   %d\n",    (int) info.st_ino);
    printf(" dev id:   %d\n",    (int) info.st_dev);
    printf("   mode:   %08x\n",        info.st_mode);
    printf("  links:   %d\n",          info.st_nlink);
    printf("    uid:   %d\n",    (int) info.st_uid);
    printf("    gid:   %d\n",    (int) info.st_gid);
    printf("  atime:   %s",     ctime(&(info.st_atime)));
    printf("  mtime:   %s",     ctime(&(info.st_mtime)));
    printf("bl size:   %d\n",    (int) info.st_blksize);
    printf(" blocks:   %d\n",    (int) info.st_blocks);
    printf("regular:   %d\n",    (int) S_ISREG(info.st_mode));
    printf("    dir:   %d\n",    (int) S_ISDIR(info.st_mode));
  }
}
```

Exemple

```
$ ./test_stat

stat(): information about /home/shak/un_truc.txt:
  inode:    1081492
 dev id:    2056
   mode:    000081a4
  links:    1
    uid:    1000
    gid:    1000
  atime:    Wed Dec  10 18:21:53 2014
  mtime:    Wed Dec  10 18:21:26 2014
bl size:    4096
 blocks:    16
regular:    1
    dir:    0
```

Rappel du plan

Exploration d'un répertoire

La consultation des répertoires est transparente vis-à-vis de l'implantation du système de fichiers au travers des quatre fonctions suivantes :

- DIR *opendir(**const char** *nom) renvoie un descripteur de répertoire lequel sera utilisé pour parcourir la liste des entrées une entrée à la fois.
- **struct** dirent *readdir(DIR *desc) renvoie l'entrée suivante du répertoire désigné par le descripteur, NULL si l'on est en fin de parcours.
- D'après la norme POSIX, une entrée de répertoire est au format suivant :

```
1 struct dirent {
2     ino_t   d_ino;        // inode de l'objet sur le disque
3     char    d_name[];     // nom de l'objet (fichier ou repertoire)
4 };
```

- L'utilisation de cette structure nécessite l'inclusion de **#include** <dirent.h> .
- **void** closedir(DIR *desc) referme le flot de lecture du répertoire, le descripteur n'est plus utilisable sauf pour le réouvrir.
- **void** rewinddir(DIR *desc) peut être utilisée afin de reprendre le parcours du répertoire désigné par descripteur au début en resynchronisant la lecture sur l'état courant du répertoire.

Rappel du plan

Exemple : mini-ls

Écrire un programme appelé `mini-ls` qui imprime des informations à propos du contenu d'un dossier donné, en utilisant :

- `opendir`
- `stat`
- `closedir`

Par défaut, il prend en argument le répertoire courant (celui d'où la commande a été lancé), sinon un répertoire :

```
$ ./minils
$ ./minils /home/shak/src/
```

mini-ls

```c
#include <stdlib.h>
#include <stdio.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <dirent.h>
#include <stdio.h>
#include <time.h>

int ls(const char* repertoire) {
  DIR           *descripteur;
  struct dirent *entree;
  struct stat info;
  descripteur = opendir(repertoire);
  if (descripteur == NULL) {
    fprintf(stderr,"Ouverture du repertoire %s impossible.\n",repertoire);
    return EXIT_FAILURE;
  }
  while ((entree = readdir(descripteur)) != NULL) {
    stat(entree->d_name, &info);
    if(S_ISDIR(info.st_mode)){
        puts(entree->d_name);
        printf("    uid:   %d\n",   (int) info.st_uid);
        printf("    gid:   %d\n",   (int) info.st_gid);
        printf("  atime:   %s",   ctime(&(info.st_atime)));
        printf("  mtime:   %s",   ctime(&(info.st_mtime)));
        printf("bl size:   %d\n",   (int) info.st_blksize);
        printf(" blocks:   %d\n",   (int) info.st_blocks);
    }
  }
  closedir(descripteur);
  return EXIT_SUCCESS;
}
```

mini-ls

```
int main(int argc,char *argv[]) {
  int i, retour;

  if (argc==1)
    retour = ls(".");
  else {
    retour = EXIT_SUCCESS;
    for (i=1; i<argc; i++){
      if (ls(argv[i]) != EXIT_SUCCESS){
        retour = EXIT_FAILURE;
      }
    }
  }
  exit(retour);
```

Questions ?