Université Paris 7
Licence 3 Informatique – Systèmes
11 janvier 2012
Durée : 3 heures. Livres et photocopies de livres interdits. Ordinateurs et téléphones portables interdits.
Les polys de cours et une feuille de notes personnelles manuscrite autorisés.

Le sujet comporte 6 pages.

Votre code doit être écrit de façon lisible, avec des indentations et les accolades appropriées permettant de voir la fin de blocs de code (fin de boucles, etc.).

Comprendre les programmes.

Exercice 1

Qu'est-ce qu'un fichier toto après l'exécution du programme suivant ?

```
1  int main(int argc, char *argv[]){
2    int d;
3    char tab[]="alamakota";
4    d = open("toto", O_WRONLY|O_TRUNC|O_CREAT|O_APPEND, S_IRUSR|S_IWUSR);
5    if(d == -1){
6      perror("open");
7      exit(EXIT_FAILURE);
8    }
9    write(d, &tab[5],4);
10   write(STDOUT_FILENO, &tab, 3);
11   dup2(d,STDOUT_FILENO);
12   write(STDOUT_FILENO, &tab[3], 2);
13   write(d,&tab,5);
14   close(d);
15   exit(EXIT_SUCCESS);
16 }
```

Qu'est-ce que ce programme affiche sur le terminal ?

Exercice 2

Qu'est-ce qu'un fichier gaga après l'exécution du programme suivant ?

```
1  int main(void){
2    int desc;
3    char *tab="abcdefghij";
4    char buf[100];
5    desc = open("gaga", O_APPEND|O_CREAT|O_RDWR|O_TRUNC, S_IRUSR|S_IWUSR);
6    if( desc <  0 ){
```

```
7       perror("open");
8       exit(1);
9     }
10    write(desc,tab,strlen(tab));
11    lseek(desc, (off_t) 0, SEEK_SET);
12    read(desc,buf,4);
13    write(desc,buf,4);
14    exit(0);
15  }
```

upposons qu a n 5 u pro ramm st r mp ac´ par

```
desc = open("gaga", O_CREAT|O_RDWR|O_TRUNC,S_IRUSR|S_IWUSR);
```

u s ra cont nu c r gaga s on x´cut pro ramm a ns mo ´?

Exercice 3

n cons `r c r toto qu s trouv ans r´p rto r /home/dupont En ut sant ls -l on trouv s propr´ta r s t s ro ts ' acc`s su vant pour home, dupont t toto

```
drwxr-xr-x   3 root root  4096 2010-09-29 20:02 home
drwx--x--x 54 dupont ens 4096 2012-01-10 09:44 dupont
-rw-------  1 dupont ens   12 2012-01-10 11:28 toto
```

n suppos qu ' ut sat ur jean st ¡r´r nt root t ¡r´r nt dupont t n appart nt pas au roup root

Question 1: L' ut sat ur jean x´cut a comman

```
cat /home/dupont/toto
```

Est-c qu c tt t ntat v ctur toto u p rm t 'a c r cont nu c c r?

Question 2: M a nt nant jean x´cut

```
ls /home/dupont
```

C a onn qu r´su tat?

Question 3: Et na m nt x´cut

```
rm /home/dupont/toto
```

Est-c qu r´uss t suppr m r toto?

Écrire des fonctions.

```
Dans toutes les programmes à écrire vous pouvez utiliser toutes les
fonctions POSIX qui conviennent à l'exception de la fonction
                        system
dont l'utilisation est prohibée.

Il est inutile d'écrire les  include.
```

Exercice 4

L but c tt x rc c st ' ´cr r un v rs on s mp ´ a com mman mv qu ´p ac s c rs auss b n or na r s qu sp´c aux ' un r´p rto r à ' autr

L ´cr tur a com mman s ra v s´ n p us urs qu st ons, c aqu qu st on man ' ´cr r un fonct on part cu `r I n' st pas n´c ssa r fa r s qu st ons ans ' or r vous ´cr v un fonct on n r´pons à a qu st on *i* vous pouv tou ours ut s r s fonct ons man ´ s ans s qu st ons pr´c´ nt s m m s vous n s av pas ncor ´cr t s Donc nut s'attar r sur un qu st on qu vous pos s prob `m s, pass p utot à a qu st on su vant t r v n p us tar s t mps p rm t

Remarques. Dans vos pro ram m s vous pouv ut s r br m nt a fonct on

```
char *concatenation(char *s,...)
```

on suppos qu' st onn´, on n man pas ' ´cr r C tt fonct on, av c nom br var ab ' ar um nts, concat`n s c a n s caract`r s pass´ s n ar um nt Par x mp

```
concatenantion("/home/dupont/bin", "/", "toto", (char *)NULL)
```

r tourn ra un c a n caract`r s `"/home/dupont/bin/toto"` Not qu rn r param`tr `concatenation` o t tr tou ours `NULL` n´c ssa r pour ´t rm n r nom br ' ar um nts

```
concatenation("toto",(char *)NULL)
```

r tourn ra ust un cop a c a n `"toto"` On suppos qu a fonct on `concatenation` r tourn ' a r ss v rs un m´mo r stat qu qu s app s su vant p uv nt mo r

D' autr s fonct ons qu p uv nt tr ut s

```
char *dirname(char *chemin);
char *basename(char *chemin);
```

`basename` r tourn rn r ´ ´m nt ' un c m n b n form´ c m n b n form´ c' st-à- r s noms s´par´s par `/`, a fonct on `dirname` r tourn c m n apr`s a suppr ss on bas nam L s ux fonct ons p uv nt mo r c m n pass´ n ar um nt onc avant s app qu r conv nt cop r c m n ans un tampon t app qu r s fonct ons sur c tampon Par x mp pour s tro s c m ns

```
char *s1="/home/dupont/toto";
char *s2="/home/toto/";
char *s3="toto";
```

`basename` r tourn ans s tro s cas `"toto"` t `dirname` r tourn r sp ct v m nt `"/home/dupont"`, `"/home"` t `"."` un c a n av c un po nt Donc n concat´nant r´su tat `dirname` app qu´ à un c m n, a c a n `"/"` t r´su tat `basename` app qu´ au m m c m n on r trouv ra un c m n ´qu va nt au c m n n t a

Dans a su t c aqu fo s quan on par c r c a v ut r c r au s ns ar c r r´ u r, r´p rto r, n symbo qu, tub nom m´ c sont sont tous s c rs on v ut par r ' un typ part cu r c r on s ra tou ours p us pr´c s

Question 1: Écr r a fonct on

```
int mv_file(char *source, char *rep)
```

C tt fonct on ´p ac c r r´ u r `source` ans r´p rto r `rep` La fonct on o t proc´ r façon su vant

- a fonct on cop c r r´ u r `source` v rs r´p rto r `rep` n ar ant nom u c r ;
- nsu t a fonct on suppr m c r r´ u r `source` ;
- a fonct on r tourn s ' op´rat on a r´uss t –´ s non

Par x mp `mv_file("/home/toto/prog.c", ".")` ´p ac c r `prog.c` u r´p rto r `/home/toto` v rs r´p rto r courant L c r cr´´ ans r´p rto r courant port ra m m nom `prog.c` On suppos ra qu s param`tr s `mv_file` sont corr ct s on n man pas v´r r s `source` st un c r r´ u r t `rep` un r´p rto r

Question 2: Écr r a fonct on

```
int mv_symlin(char *source, char *rep)
```

C tt fonct on ´p ac n sym bo qu `source` v rs r´p rto r `rep` La fonct on proc` façon su vant

- on t cont nu u n sym bo qu `source` t on cr´ ans `rep` un n sym bo qu av c m m nom t m m cont nu qu `source` ;
- on suppr m `source` c' st m m app syst`m qu suppr m s c rs r´ u rs ,
- on r tourn n cas succ`s t –´ s ´p ac m nt ´c ou

Donc `mv_symlin("/home/toto/coto", "./bin")` cr´ ans r´p rto r `./bin` un n sym bo qu ont nom st `coto` t ont cont nu st m m qu cont nu u n `/home/toto/coto` Comm ans a qu st on pr´c´ nt on assum qu s param`tr s `mv_symlinl` sont corr ct s t on n man pas s v´r r

Question 3: Écr r a fonct on

```
int mv_fifo(cher *source, char *rep)
```

L param`tr `source` st un c m n v rs un tub nomm´ La fonct on cr´ ans r´p rto r `rep` un tub nomm´ m m nom qu tub r´f´r nc´ par `source` t suppr m tub `source` pour a suppr ss on tub s nomm´s on ut s a m m fonct on qu st ut s´ pour suppr m r s c rs r´ u rs

Question 4: Écr r a fonct on

```
int type_fichier(char *chemin)
```

qu r tourn

a s `chemin` st un r´p rto r ,

b –´ s `chemin` st un c r r´ u r ;

c s `chemin` st un n sym bo qu ,

3 s `chemin` st un tub nomm´ ;

–´ s aucun cas pr´c´ ant n' st va

Question 5: Écr r a fonct on

```
int meme(char *premier, char *deuxieme)
```

C tt fonct on r tourn ra

a s `premier` t `deuxieme` sont s c m ns v rs m m c r ;

b ′ s `premier` t `deuxieme` sont s c m ns v rs s c rs ¡ ´r nts ma s s ux c rs r´s nt ans m m syst`m c rs c´ st-à- r m m vo um o qu , ou a m m part t on ,

c s s ux c rs r´s nt ans s syst`m s c rs ¡ ´r nts ;

—′ s on n p ut pas v´r r où r´s un c s c rs su t à un app à `lstat` qu a ´c ou´

Rappel. L m m c r c a v ut r m m nu m´ro -no t m m vo um o qu m m syst`m c r , par contr `cg` t `cd` p uv nt tr comp`t nt ¡ ´r nts n tant qu c a n s caract`r s

Question 6:

Préambule. L s fonct ons qu vous av u s à ´cr r ans s qu st ons ′, , 3 sont nut m nt comp qu´ s s nous vou ons ´p ac r un c r s map [1]

ntr ux r´p rto r s qu ans m m syst`m c rs ans a m m part t on o qu Par x map s nous vou ons ´p ac r un c r s map `/home/toto/ff` v rs r´p rto r `./bin` t s ux r´p rto r s `/home/toto` t `./bin` r´s nt ans a m m part t on vo r a fonct on pr´c´ nt a ors su t

a cr´ r ans `./bin` un n ur ont nom st `ff` t qu r´f´r nc m m c r qu `/home/toto/ff` ;

b t nsu t suppr m r `/home/toto/ff`

Donc tout s r´sum à un cr´at on ' un nouv ntr´ ans r´p rto r `./bin` t a suppr ss on ' un ntr´ r´p rto r `/home/toto` un s map app à `link` su v ' un app à `unlink` pas b so n cr´ r un cop p ys qu c r

Fin de préambule.

Écr r a fonct on

```
int mv_tout(char *source, char *rep)
```

qu ´p ac `source` v rs `rep` n map´m ntant ' a or t m su vant

′ s `rep` n' st pas un r´p rto r a ors a fonct on r tourn m m´ at m nt —′ ,

s `source` st un c r s map c r r´ u r , n sym bo qu ou un tub nom m´ t r´p rto r « `dirname` sourc » t r´p rto r « `rep` » r´s nt s tous s ux ans a m m part t on pour v´r r on ut s ra a fonct on a qu st on pr´c´ nt a ors on ´p ac `source` v rs `rep` n ut sant a m´t o ´cr t ans pr´ambu t a fonct on r tourn ′ ,

3 s `source` st un c r s map c r r´ u r , n sym bo qu ou un tub nom m´ t s r´p rto r s « `dirname source` » t « `rep` » r´s nt ans s part t ons ¡ ´r nt s a ors on ´p ac `source` v rs `rep` n ut sant un tro s fonct ons s qu st ons ′, , 3 s on cas

1. Ici et par la suite un « fichier simple » signifie que c'est soit un fichier régulier, soit un lien symbolique, soit un tube nommé. Un répertoire n'est pas un fichier simple.

s `source` st un r´p rto r a ors on ´p ac r´curs v nt tous s c rs t sous-r´p rto r s `source` v rs `rep` n ut sant ’a or t su vant

on parcourt r´p rto r `source` t pour c aqu ntr´ `XXX` c r´p rto r r´ nt . t . .

a s `source/XXX` st un c r s p a ors on a t app r´curs f `mv_tout(source/XXX, rep)` `source/XXX` ´s n a concat´nat on c a n s « `source` », `"/"` t « `XXX` »,

b s `source/XXX` st un r´p rto r a ors
- on cr´ un sous-r´p rto r `XXX` ans `rep`,
- nsu t on a t un app r´curs f à `mv_tout(source/XXX, rep/XXX)`,
- t na nt on suppr r´p rto r `rep/XXX`

Dans s cas , 3, a fonct on `mv_tout` r tourn r