SY5 – Systèmes d'exploitation

1re session
Examen de substitution – 12 février 2022

Durée : 2 heures

Aucun document autorisé excepté une feuille A4 manuscrite
Appareils électroniques éteints et rangés

Tous les programmes demandés doivent être écrits en C, de la manière la plus lisible possible, c'est-à-dire ***bien indentés*** *et* ***commentés*** *aux endroits où cela paraît nécessaire.*

En revanche, s'agissant d'une épreuve sur papier, il ne vous est pas demandé un code irréprochable ; en particulier il est inutile d'indiquer les inclusions nécessaires, et vous pouvez vous contenter d'une gestion minimale des erreurs (par exemple en utilisant la fonction `assert()`*, ou même simplement avec un commentaire* `/* erreur à gérer */`*).*

Exercice 1 : histoires de famille

On considère une commande « `zombies` » dont l'exécution provoque la situation suivante :

```
poulalho@lulu:SY5$ ./zombies &
poulalho@lulu:SY5$ sleep 5; ps j
   PPID     PID    PGID     SID TTY      STAT   UID   TIME COMMAND
1952538 1952539 1952539 1952539 pts/20   Ss    8159   0:00 -bash
1952539 1963818 1963818 1952539 pts/20   S     8159   0:00 zombies
1963818 1963822 1963818 1952539 pts/20   S     8159   0:00 zombies
1963822 1963823 1963818 1952539 pts/20   S     8159   0:00 zombies
1963823 1963824 1963818 1952539 pts/20   Z     8159   0:00 [sleep] <defunct>
1963823 1963825 1963818 1952539 pts/20   Z     8159   0:00 [sleep] <defunct>
1963823 1963826 1963818 1952539 pts/20   Z     8159   0:00 [sleep] <defunct>
1952539 1963861 1963861 1952539 pts/20   R+    8159   0:00 ps j
poulalho@lulu:SY5$ sleep 5; ps j
   PPID     PID    PGID     SID TTY      STAT   UID   TIME COMMAND
1952538 1952539 1952539 1952539 pts/20   Ss    8159   0:00 -bash
1952539 1963818 1963818 1952539 pts/20   S     8159   0:00 zombies
1963818 1963822 1963818 1952539 pts/20   S     8159   0:00 zombies
1952539 1963932 1963932 1952539 pts/20   R+    8159   0:00 ps j
```

1. Dessiner la généalogie des processus créés par « `zombies` ».
2. Expliquer précisément l'état de chacun au bout de 5 secondes.
3. Que s'est-il passé durant les 5 secondes suivantes ?
4. Écrire un programme `zombies.c` ayant le comportement observé.

Exercice 2 : autour des tampons de la bibliothèque standard

On considère les deux programmes suivants :

```
 /*  pif.c  */
int main() {
  printf("pif ");
  sleep(1);
  printf("paf\n");
  sleep(1);
  printf("pouf");
  exit(0);
}
```

```
 /*  _pif.c  */
int main() {
  printf("pif ");
  sleep(1);
  printf("paf\n");
  sleep(1);
  printf("pouf");
  _exit(0);
}
```

On exécute les deux programmes via « `strace` », en redirigeant la sortie soit vers un terminal, soit vers un fichier ordinaire.
Déterminer quelles sont les traces possibles parmi les traces présentées en dernière page ; donner un exemple de ligne de commande produisant chaque trace possible, et expliquer pourquoi les autres ne le sont pas.

Exercice 3 : parcours de répertoires

Dans cet exercice, il est interdit d'exécuter une autre commande que le programme à écrire (en particulier, pas de « `find` »), et d'utiliser les bibliothèques `ftw` et `fts`.

1. Écrire un programme `gros.c` qui parcourt le répertoire courant et liste les entrées correspondant à des fichiers ordinaires de plus d'un gigaoctet (2^{30} octets).
2. Modifier ce programme pour qu'il fasse de même dans toute *l'arborescence* dont le répertoire courant est la racine.

(On supposera que l'arborescence ne contient pas de liens symboliques)

Exercice 4 : nombre de fils d'un processus

On rappelle que l'option « `--ppid` » de « `ps` » permet de filtrer les processus affichés en fonction de leur processus parent ; par exemple dans le terminal d'où a été lancée la commande « `zombies` » à l'exercice 1, cela donne (`$$` désigne le pid du shell courant) :

```
poulalho@lulu:SY5$ ps --ppid $$
    PID TTY          TIME CMD
1963818 pts/20   00:00:00 zombies
1967036 pts/20   00:00:00 ps
```

Écrire un programme `compte_fils.c` prenant un argument et utilisant « `ps` » pour compter les processus fils du processus dont l'identifiant est passé en paramètre. Votre programme ne doit exécuter aucune autre commande que « `ps` ». L'usage de la bibliothèque `stdio` doit être limité à la gestion des erreurs et à l'affichage final.

Exercice 5 : un problème de synchronisation

1. Quels affichages le programme suivant peut-il produire ?

```
int main() {
  if (fork() == 0) printf("ping ");
  else if (fork() == 0) printf("pong ");
  else printf("pang ");
}
```

2. Proposer une modification minimale permettant d'assurer que l'affichage soit toujours [1] `"ping pong pang "`.

3. Cette solution est-elle adaptable pour assurer un autre affichage (sans changer le message affiché par chaque processus) ?

4. Proposer une solution pour assurer l'affichage `"pang pong ping "`, sans changer l'ordre des `fork()` et des `printf()`, en synchronisant les processus à l'aide de tube(s) anonyme(s).

5. Proposer une autre solution de synchronisation reposant sur l'envoi de signaux. Expliquer en particulier quand le(s) gestionnaire(s) doi(ven)t être mis en place. S'il y a un risque de blocage, expliquer comment l'éviter.

Petit mémento

```
ssize_t read(int fd, void *buf, size_t count);
ssize_t write(int fd, const void *buf, size_t count);
DIR *opendir(const char *name);
struct dirent *readdir(DIR *dirp);
int closedir(DIR *dirp);
struct dirent {    /* contient entre autres : */
    ino_t  d_ino;      /* Inode number */
    char   d_name[];   /* Null-terminated filename */
};
int stat(const char *pathname, struct stat *statbuf);
struct stat {   /* contient entre autres : */
    dev_t    st_dev;       /* ID of device containing file */
    ino_t    st_ino;       /* Inode number */
    mode_t   st_mode;      /* File type and mode */
    off_t    st_size;      /* Total size, in bytes */
};
int execlp(const char *file, const char *arg, ... /* (char *) NULL */);
int execvp(const char *file, char *const argv[]);
int kill(pid_t pid, int sig);
int sigaction(int signum, const struct sigaction *act, struct sigaction *oldact);
struct sigaction {  /* contient entre autres : */
  void    (*sa_handler)(int);
  sigset_t  sa_mask;
  int       sa_flags;
}
```

1. sauf en cas d'erreur de `fork()`, naturellement

Exercice 2 : autour des tampons de la bibliothèque standard

1. a.
```
[...]
fstat(1, {st_mode=S_IFCHR|0620, [...]}) = 0
[...]
clock_nanosleep([...], tv_sec=1, [...]) = 0
write(1, "pif ", 4)                     = 4
clock_nanosleep([...], tv_sec=1, [...]) = 0
write(1, "paf\n", 4)                    = 4
clock_nanosleep([...], tv_sec=1, [...]) = 0
write(1, "pouf", 4)                     = 4
exit_group(0)                           = ?
+++ exited with 0 +++
```

b.
```
[...]
fstat(1, {st_mode=S_IFCHR|0620, [...]}) = 0
[...]
clock_nanosleep([...], tv_sec=1, [...]) = 0
write(1, "pif paf\n", 8)                = 8
clock_nanosleep([...], tv_sec=1, [...]) = 0
write(1, "pouf", 4)                     = 4
exit_group(0)                           = ?
+++ exited with 0 +++
```

c.
```
[...]
fstat(1, {st_mode=S_IFCHR|0620, [...]}) = 0
[...]
clock_nanosleep([...], tv_sec=1, [...]) = 0
clock_nanosleep([...], tv_sec=1, [...]) = 0
write(1, "pif paf\npouf", 12)           = 12
exit_group(0)                           = ?
+++ exited with 0 +++
```

d.
```
[...]
fstat(1, {st_mode=S_IFCHR|0620, [...]}) = 0
[...]
clock_nanosleep([...], tv_sec=1, [...]) = 0
write(1, "pif paf\n", 8)                = 8
clock_nanosleep([...], tv_sec=1, [...]) = 0
exit_group(0)                           = ?
+++ exited with 0 +++
```

e.
```
[...]
fstat(1, {st_mode=S_IFCHR|0620, [...]}) = 0
[...]
clock_nanosleep([...], tv_sec=1, [...]) = 0
clock_nanosleep([...], tv_sec=1, [...]) = 0
exit_group(0)                           = ?
+++ exited with 0 +++
```

2. a.
```
[...]
fstat(1, {st_mode=S_IFREG|0644, st_size=0, ...}) = 0
[...]
clock_nanosleep([...], tv_sec=1, [...]) = 0
write(1, "pif ", 4)                     = 4
clock_nanosleep([...], tv_sec=1, [...]) = 0
write(1, "paf\n", 4)                    = 4
clock_nanosleep([...], tv_sec=1, [...]) = 0
write(1, "pouf", 4)                     = 4
exit_group(0)                           = ?
+++ exited with 0 +++
```

b.
```
[...]
fstat(1, {st_mode=S_IFREG|0644, st_size=0, ...}) = 0
[...]
clock_nanosleep([...], tv_sec=1, [...]) = 0
write(1, "pif paf\n", 8)                = 8
clock_nanosleep([...], tv_sec=1, [...]) = 0
write(1, "pouf", 4)                     = 4
exit_group(0)                           = ?
+++ exited with 0 +++
```

c.
```
[...]
fstat(1, {st_mode=S_IFREG|0644, st_size=0, ...}) = 0
[...]
clock_nanosleep([...], tv_sec=1, [...]) = 0
clock_nanosleep([...], tv_sec=1, [...]) = 0
write(1, "pif paf\npouf", 12)           = 12
exit_group(0)                           = ?
+++ exited with 0 +++
```

d.
```
[...]
fstat(1, {st_mode=S_IFREG|0644, st_size=0, ...}) = 0
[...]
clock_nanosleep([...], tv_sec=1, [...]) = 0
write(1, "pif paf\n", 8)                = 8
clock_nanosleep([...], tv_sec=1, [...]) = 0
exit_group(0)                           = ?
+++ exited with 0 +++
```

e.
```
[...]
fstat(1, {st_mode=S_IFREG|0644, st_size=0, ...}) = 0
[...]
clock_nanosleep([...], tv_sec=1, [...]) = 0
clock_nanosleep([...], tv_sec=1, [...]) = 0
exit_group(0)                           = ?
+++ exited with 0 +++
```

3. a.
```
[...]
fstat(3, {st_mode=S_IFREG|0644, st_size=0, ...}) = 0
[...]
clock_nanosleep([...], tv_sec=1, [...]) = 0
write(3, "pif ", 4)                     = 4
clock_nanosleep([...], tv_sec=1, [...]) = 0
write(3, "paf\n", 4)                    = 4
clock_nanosleep([...], tv_sec=1, [...]) = 0
write(3, "pouf", 4)                     = 4
exit_group(0)                           = ?
+++ exited with 0 +++
```

b.
```
[...]
fstat(3, {st_mode=S_IFREG|0644, st_size=0, ...}) = 0
[...]
clock_nanosleep([...], tv_sec=1, [...]) = 0
write(3, "pif paf\n", 8)                = 8
clock_nanosleep([...], tv_sec=1, [...]) = 0
write(3, "pouf", 4)                     = 4
exit_group(0)                           = ?
+++ exited with 0 +++
```

c.
```
[...]
fstat(3, {st_mode=S_IFREG|0644, st_size=0, ...}) = 0
[...]
clock_nanosleep([...], tv_sec=1, [...]) = 0
clock_nanosleep([...], tv_sec=1, [...]) = 0
write(3, "pif paf\npouf", 12)           = 12
exit_group(0)                           = ?
+++ exited with 0 +++
```

d.
```
[...]
fstat(3, {st_mode=S_IFREG|0644, st_size=0, ...}) = 0
[...]
clock_nanosleep([...], tv_sec=1, [...]) = 0
write(3, "pif paf\n", 8)                = 8
clock_nanosleep([...], tv_sec=1, [...]) = 0
exit_group(0)                           = ?
+++ exited with 0 +++
```

e.
```
[...]
fstat(3, {st_mode=S_IFREG|0644, st_size=0, ...}) = 0
[...]
clock_nanosleep([...], tv_sec=1, [...]) = 0
clock_nanosleep([...], tv_sec=1, [...]) = 0
exit_group(0)                           = ?
+++ exited with 0 +++
```