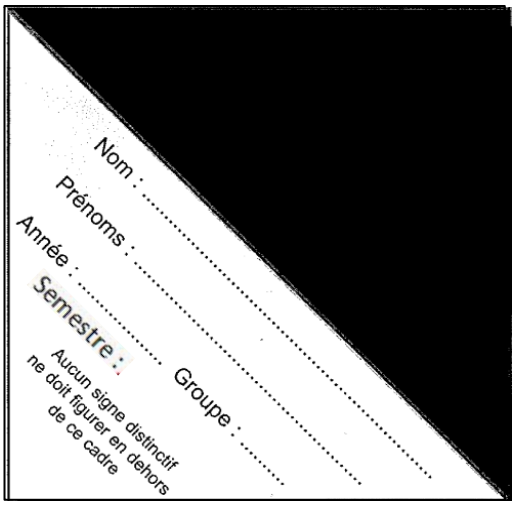
NOTE

UNIVERSITÉ
NICE SOPHIA ANTIPOLIS
INSTITUT UNIVERSITAIRE DE TECHNOLOGIE
DEPARTEMENT INFORMATIQUE



Épreuve : S3T - M311 – IE *(Coefficient : 0,75 / 2,5)*
Date : Jeudi 15 novembre 2018
Durée : 1h30 minutes
Conditions de l'épreuve : Aucun appareil électronique, aucun document n'est autorisé … Répondre directement sur cette copie et utiliser si besoin des intercalaires. Des points seront retirés en cas d'écriture illisible.

…… REPONDRE A 3 QUESTIONS SUR 4 ……

Question 1 : 7 points – Programmation avec l'API « fichiers » Répondre dans le cadre

1.1 En utilisant en particulier les routines `assert` **et (** `opendir` **ou** `chdir` **) écrire le programme** C **décrit ci-dessous :**

Synopsis: **IEQ1 <chemin> ...**

Rôle: recherche la présence de chaque <chemin> dans l'arborescence et affiche sur la sortie standard si ce <chemin> désigne ou non un répertoire accessible.

Exemple - la séquence suivante …

```
mkdir ESSAI ; chmod 000 ESSAI
IEQ1 /etc /etc/passwd .. ESSAI
```

… Donne

/etc : est un répertoire accessible
/etc/passwd : n'est pas un répertoire accessible
.. : est un répertoire accessible
ESSAI : n'est pas un répertoire accessible

Cas d'erreur à traiter:

- Moins de 2 mots dans la ligne de commande
- Accès impossible à <chemin>

Répondre sur la dernière page si besoin.

Ne rien écrire dans ce cadre

Question 2 : 7 points – Programmation avec fork et execlp — Répondre dans le cadre

Soit le programme `imprime.c` **(et son executable** `imprime`**) suivant :**

```
/// #include omis

int main(int argc, char *argv[])
{
    assert (argc == 3);
    int i, nombre = atoi (argv[1]);
    for (i = 0; i < nombre; i++)
        printf("%c", argv[2][0]);
    return 0;
}
```

2.1 Ecrire un programme C **qui met en œuvre 3 processus en parallèle. Ce programme :**

- prend en paramètre une chaine de caractères supposée représenter un nombre entier noté <N> **dans la suite.**

- engendre un fils qui lui-même engendre un fils.

Chacun des processus affiche <N> **caractères identiques en utilisant <u>obligatoirement</u> l'exécutable** `imprime` **(ci-dessus) :** `'/'` **pour le père,** `'*'` **pour le fils et** `'.'` **pour le petit-fils.**

Les cas d'erreur à prendre en compte sont l'absence d'argument et l'échec de la création d'un processus.

Répondre sur la dernière page si besoin.

Question 3 : 7 points - Pipelines et redirections — Répondre dans le cadre

Soit le programme ci-dessous à compiler avec `gcc duppipe.c –o duppipe`

```
/// #include omis

int main (int argc, char* argv[]) {
   int p[2];
   assert (argc == 3);
   assert (pipe (p) != -1);
   if (fork () > 0)
    { close (1); dup (p[1]); close (p[0]); close (p[1]);
      execlp (argv[1], argv[1], NULL);      /* POINT_A */}
   else
    { close (0); dup (p[0]); close (p[0]); close (p[1]);
      execlp (argv[2], argv[2], NULL);      /* POINT_B */}
}
```

Questions :

3.1 On se place juste après la réussite des deux routines `execlp` **(après** `POINT_A` **et** `POINT_B`**) : dessiner et commenter le plus précisément possible les tables des descripteurs de chaque processus et leurs liaisons avec la table des fichiers ouverts (comme présenté en cours).**

Répondre sur la dernière page si besoin.

Question 4 : 7 points - Programmation signaux — Répondre dans le cadre

Soit le programme (incomplet) ci-dessous qui intercepte des signaux. Dans sa version complète, il doit faire ce qui est indiqué dans les commentaires.

```
/// #include omis

/// Traitement du signal SIGSEGV
/// A la première reception du signal, s'envoie SIGUSR2
/// A la deuxieme reception du signal, termine le processus

/// Traitement du signal SIGUSR2
/// A la première reception du signal, remet le comportement par défaut pour ce
/// signal

int main(){
 /// Detourner le signal SIGSEGV vers une routine de traitement du signal SIGSEGV
 /// Detourner le signal SIGUSR2 vers une routine de traitement du signal SIGUSR2
  while (getchar() != 'F');
 }
```

4.1 Donner le code C correspondant à ce qui est à faire. Donner toutes les explications nécessaires pour être compréhensible.

UNIVERSITE DE NICE SOPHIA ANTIPOLIS — IUT NICE – COTE D'AZUR

Département INFORMATIQUE – D.U.T. INFORMATIQUE

Modules M311

Interrogation écrite – SUPPORT autorisé 2018-2019

API FICHIERS

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

int open(const char *pathname, int flags, mode_t mode);
// flags est l'un des éléments O_RDONLY, O_WRONLY  ou  O_RDWR

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 *nom);

struct dirent *readdir(DIR *dir);
Avec
struct dirent {
   ino_t          d_ino;       /* numéro de l'inode                 */
   off_t          d_off;       /* décalage vers le prochain dirent  */
   unsigned short d_reclen;    /* longueur de cet enregistrement    */
   unsigned char  d_type;      /* type du fichier                   */
   char           d_name[256]; /* nom du fichier                    */
};

int stat(const char *path, struct stat *buf);
Avec
struct stat {
   dev_t     st_dev;     /* Périphérique                       */
   ino_t     st_ino;     /* Numéro i-nœud                      */
   mode_t    st_mode;    /* Protection                         */
   nlink_t   st_nlink;   /* Nb liens matériels                 */
   uid_t     st_uid;     /* UID propriétaire                   */
   gid_t     st_gid;     /* GID propriétaire                   */
   dev_t     st_rdev;    /* Type périphérique                  */
   off_t     st_size;    /* Taille totale en octets            */
   blksize_t st_blksize; /* Taille de bloc pour E/S            */
   blkcnt_t  st_blocks;  /* Nombre de blocs alloués            */
   time_t    st_atime;   /* Heure dernier accès                */
   time_t    st_mtime;   /* Heure dernière modification        */
   time_t    st_ctime;   /* Heure dernier changement état      */
 };
Et quelques macros POSIX (sur le champ st_mode) :

           S_ISREG(m)  un fichier ordinaire ?
           S_ISDIR(m)  un répertoire ?
           S_ISLNK(m)  un lien symbolique ? (Pas dans POSIX.1-1996).

int chdir(const char *path);
```

API PROCESSUS

```
#include <unistd.h>

pid_t fork(void);

int execl(const char *path, const char *arg, ...);

int execlp(const char *file, const char *arg, ...);

int execle(const char *path, const char *arg, ..., char * const envp[]);

int execv(const char *path, char *const argv[]);

int execvp(const char *file, char *const argv[]);

void exit(int status);
```

API PIPE-REDIR

```
#include <unistd.h>

int dup(int oldfd);

int dup2(int oldfd, int newfd);

int pipe(int pipefd[2]);
```

API SIGNAUX (éléments de ...)

```
#include <unistd.h>
#include <sys/types.h>
#include <sys/time.h>
#include <sys/resource.h>
#include <sys/wait.h>
#include <signal.h>

unsigned int sleep(unsigned int nb_sec);

pid_t wait(int *status);

pid_t wait3(int *status, int options, struct rusage *rusage);
// Exemple TP : wait3 (NULL, WNOHANG, NULL);

sighandler_t signal(int signum, sighandler_t handler);
Avec typedef void (*sighandler_t)(int);
```

API CHAINES

```
#include <stdio.h>   #include <string.h>  #include <strings.h>

char *strcpy(char *dest, const char *src);
char *strcat(char *dest, const char *src);
char *strtok(char *str, const char *delim); // str = NULL au 2ème appel
int sprintf(char *str, const char *format, ...);
void bzero(void *s, size_t n);
void bcopy(const void *src, void *dest, size_t n)
```