Proc — PDS — TD 1 du 15/20

Exercice 1:

Q2: A: → je suis Pe ?

Pour la suite, on ne peut pas savoir car les deux s'exécutent en //

Q3: A:

je suis le fils

B:

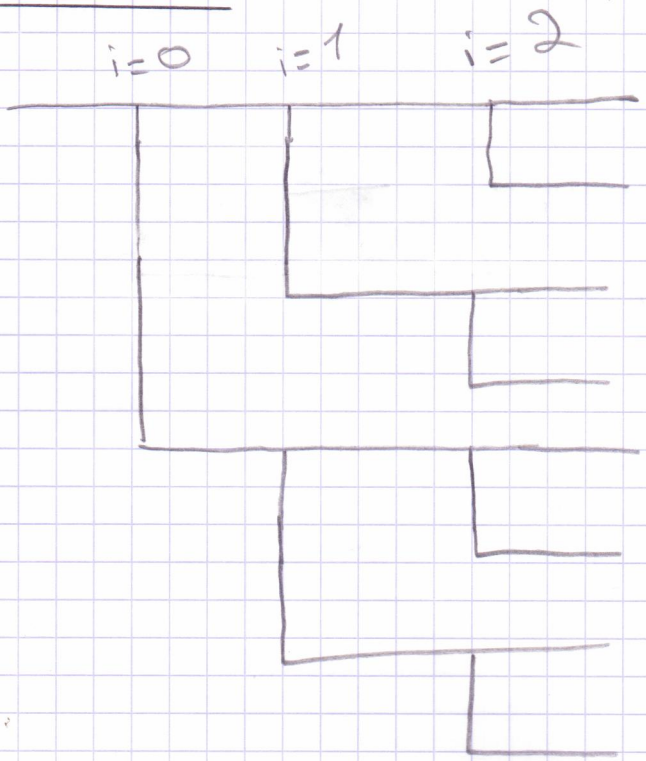
je suis le père

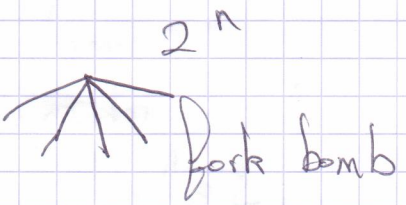
B:

Q4: Gestion d'erreur

Q5: vider le tampon d'appel système printf

Exercice 2:





Q3: ~~pas de~~ wait / ~~pas de~~ fflush

Exercice 4:

```
void compte ( n=3 ){
    if(n ==0) return;
    pid_t pid;
    pid = fork();
    if ( pid != 0) {
        wait (NULL);
        printf(n); fflush();
    } else if (pid==0) {
        compte (n-1);
        return;
    } else {
        perror("erreur fork", exit(EXIT_FAILURE);
}
```

Exercice 3 :

Q1/2 :

```
int main(void){
    int i;
    for(int i=0; i<3; i++){
        switch(fork()){
            case -1: perror("Erreur Fork"), exit(EXIT_FAILURE);
            case 0: fprintf("affiche %d\n", i+1);
                    3-(i+1)

            default: fprintf("affiche %d\n", i);
                     3-i
                     exit(1);
                     break
        }
    }
    wait(NULL);
    exit(1);
}
```

Exercice 4 :

```
tri f(f1,f2,f3){
    int i;
    for(i=0; i<2; i++){
        switch(fork()){
            case -1: perror("erreur fork");exit(Exit_FAILURE);
            case 0:
```

Exo 4+5 : Exo 5

```
void trif (f1, f2, f3) {
```
funct_t f[], char * args[], int n) {

on oublie
```
    typedef void (*fct_t)(void);
    fct_t tab[3] = {f1, f2, f3}; /* C99 ou C89 */
    int i;
    for (i=0; i<3 ; i++) {
        switch (fork()) {
            case -1: /* gestion d'erreur */
            case 0: tab[i]; exit(0);
                    tab[i](args);
            default:
    }
    for (i=0; i<3 ; i++)
        wait(NULL);
```
n

PDS TD ⑨ du 22/10

Exo 9

```
int main (int argc, char **argv){
  int i; char *str; int status;
  assert ((pid = fork()) != -1);
  if (pid == 0){
    for (i = 0; !strcmp(argv[i], "do"); i++);
    argv[i] = NULL;
    str = argv[1];
    assert (execvp(str, argv) != -1);
  }
  else if (pid != -1){
    wait (&status);
    if (WIFEXITED (status) &&
        WIFSTATUS(status) == EXIT_SUCCESS){
      str = argv[i+1];
      argv += i+2;
      assert (execvp(str, argv) != -1);
  }
  return -1; // should never happen
}
```

attention à l'overflow ←

Exo 10

```
int main (int argc, char ** argv) {
    int fd; int i; char * cmd;

    fd = open (argv[1], O_RDWR | O_CREAT | O_TRUNC
    assert (fd != -1);
    dup2 (fd, 1);
    close (fd);

    for (i = 2 ; i < argc; i++) {
        cmd = argv[i];
        switch (fork()) {
            case -1: error
            case 0: execl (cmd, cmd, NULL)
                    return -1; // should never happen
            default: wait (NULL);
        }
    }
    return 1;
}
```