Enoncé à rendre à la fin !! Mettez votre nom, et cochez l'avancement de chaque exercice

Recommandations : c'est du système, pensez bien aux tests des primitives. Évitez les mallocs, inutile de complexifier. En cas de besoin, servez vous du man.

1) : Unstoppable count (3 pts)

Ecrire un programme qui fait patienter 10 secondes en affichant tous les chiffres de 0 à 9 (un par seconde, voir la fonction sleep()). Ce programme ne doit pas pouvoir être interrompu par CTRL C !

```c
#include<unistd.h>
#include<stdio.h>
#include<signal.h>

void nein(int num) {
    printf("Aie !\n");
}

int main(int argc, char ** argv) {
    int i;
    signal(SIGINT,nein);
    for(i=0;i<10;i++) {
        printf("%d\n",i+1);
        sleep(1);
    }
    return 0;
}
```

2) Vice et Versa(3 pts)

Ecrire un programme prenant en argument deux noms de fichiers. Si ces deux fichiers correspondent à la même inode, il affiche « meme inode » suivi du numéro d'inode et la taille du contenu. Sinon, il affiche les deux noms de fichiers suivi de leur numéro d'inode et la taille des contenu. Ce programme retourne 0 si l'inode est identique, 1 sinon.

```c
#include<stdio.h>
#include<unistd.h>
#include<stdlib.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>


int main(int argc, char** argv) {
    struct stat st1, st2;
    if (3 != argc) {
        fprintf(stderr,"I need two files to compare\n");
        return EXIT_FAILURE;
    }
    if(-1 == stat(argv[1], &st1)) {
        perror("Can't handle the first file");
        return EXIT_FAILURE;
    }
    if(-1 == stat(argv[2], &st2)) {
        perror("Can't handle the second file");
        return EXIT_FAILURE;
    }
    if (st1.st_ino == st2.st_ino) {
        printf("Same Inode %d\n", st1.st_ino);
    } else {
        printf("Different inodes\n%s : %d\n%s : %d\n",argv[1], st1.st_ino argv[2], st2.st_ino);
    }

    return EXIT_SUCCESS;
}
```

3) : écrire la commande `tee`.(4 pts)

La commande **`tee`** copie l'entrée standard sur la sortie standard ainsi que dans tous les fichiers passés en paramètre.
Exemple (ici, avec une seule ligne, mais l'entrée standard pourra être bien plus importante):

```
$ date | tee save_date toto
Wed Mar 30 17:12:12 CEST 2020
$ cat save_date
Wed Mar 30 17:12:12 CEST 2020
$ cat toto
Wed Mar 30 17:12:12 CEST 2020
```

```c
#include<stdio.h>
#include<unistd.h>
#include<stdlib.h>
#include <fcntl.h>


int main(int argc, char** argv) {
    int fd[argc-1], i, nblus;
    char buffer[1024];

    for(i=0;i<argc-1;i++) {
        fd[i]=open(argv[i+1],O_WRONLY | O_CREAT | O_TRUNC,0644);
        if (fd[i] == -1) {
            perror("Issue with open ?");
            return EXIT_FAILURE;
        }
    }

    while((nblus=read(0,buffer,1024))>0) {
        for (i=0;i<argc-1;i++) {
            if (-1 == write(fd[i],buffer,nblus)) {
                perror("Issue with write ??");
                return EXIT_FAILURE;
            }
        }
        write(1,buffer,nblus);
    }
    for(i=0;i<argc-1;i++) {
        if (-1 == close(fd[i])) {
            perror("Issue with close ??");
            return EXIT_FAILURE;
        }
    }
    return EXIT_SUCCESS;
}
```

4) : 1337(6 pts)

Ecrire **leet**, le programme capable de lancer une commande et toutes ses options passées en argument de la ligne de commande, et qui transforme sa sortie au format *leet* (S et s deviennent 5, A et a deviennent 4, E et e => 3, 0 et o => 0, T et t => 7, I et i =>1).

Voir l'exemple ci-dessous

```
sylvain@ilium:~$ ./leet who
5ylv41n  77y2        2013-11-02 08:57 (:0)
5ylv41n  p75/1       2013-11-02 16:02 (:0)
5y1v4in  p75/2       2013-11-02 16:18 (:0)
```

```
sylvain@ilium:~$ ./leet ls -l /tmp
-rw-rw-r--  1 5ylv41n 5ylv41n 1242  n0v.  10  2012  m4573r.c
-rwxr-xr-x 1 5ylv41n 5ylv41n 8622  0c7.  28 14:45  pr073c73dG3d17
-rw-rw-r--  1 5ylv41n 5ylv41n 1743  n0v.  10  2012  50ld13r.c
-rw-r--r--  1 5ylv41n 5ylv41n 1016   n0v.   2 17:23 5uch45h3ll.c
-rwxr-xr-x 1 5ylv41n 5ylv41n 9210   0c7. 28 14:30  71m3F0rk
```

(Les fichier dans /tmp sont *master.c, protectedGedit, soldier.c, suchAShell.c et TimeFork, ils appartiennent à Sylvain (5ylv41n)*)

```c
#include<unistd.h>
#include <sys/types.h>
#include <sys/stat.h>
#include<stdio.h>
#include <string.h>
#include<stdlib.h>
#include <fcntl.h>


int main(int argc, char ** argv) {
    if (1==argc) {
            fprintf(stderr,"Usage : leet cmd [args ...] \n");
            return EXIT_FAILURE;
        }

    int p[2];
    char c; //lecture du caractère envoyé dans le tube

    if(-1==pipe(p)) {
        perror("plantage du tuyau");
        return EXIT_FAILURE;
    }

    switch(fork()) {
        case -1 : perror("plantage du fork");
                  return EXIT_FAILURE;
        case 0 : //je suis le fils, je lance la commande en argv[1]
            close(p[0]);
            dup2(p[1],1);
            close(p[1]);
            execvp(argv[1],argv+1);
            perror("Cette commande n'existe pas");
            return EXIT_FAILURE;
```

```c
        default :
            close(p[1]);
            while(0!=read(p[0],&c,1)) {
                switch(c) {
                case 'e' :
                case 'E' :
                    printf("3");
                    break;
                case 's' :
                case 'S' :
                    printf("5");
                    break;
                case 'a' :
                case 'A' :
                    printf("4");
                    break;
                case 'i' :
                case 'I' :
                    printf("1");
                    break;
                case 'o' :
                case 'O' :
                    printf("0");
                    break;
                default : printf("%c",c);
                }
            }

    }
    close(p[0]);
    return EXIT_SUCCESS;
}
```

5) El calculator (4 pts)

Ecrire un programme qui lance 10 processus, chacun d'eux affichant une table de multiplication (le premier processus est chargé de la table des 1, le second de la table des 2, etc). Lorsque tout est terminé, le message : « c'est terminé » est affiché.

```c
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <unistd.h>


int main(int argc,char* argv[]) {

for(int i=1; i<11;i++) {
        switch (fork()) {
                case -1: perror("fork"); exit(1);
                case 0:
                        printf("\n\nEl calculator : Table de %d\n", i);
                        for (int j=1; j<11; j++) {
                                printf("%d * %d = %d\n", j, i, j*i);
                        }
                        return EXIT_SUCCESS;
                default:
                        waitpid(-1,NULL,0);
        }
}
printf("\n\nC'est terminé\n");
return EXIT_SUCCESS;
}
```