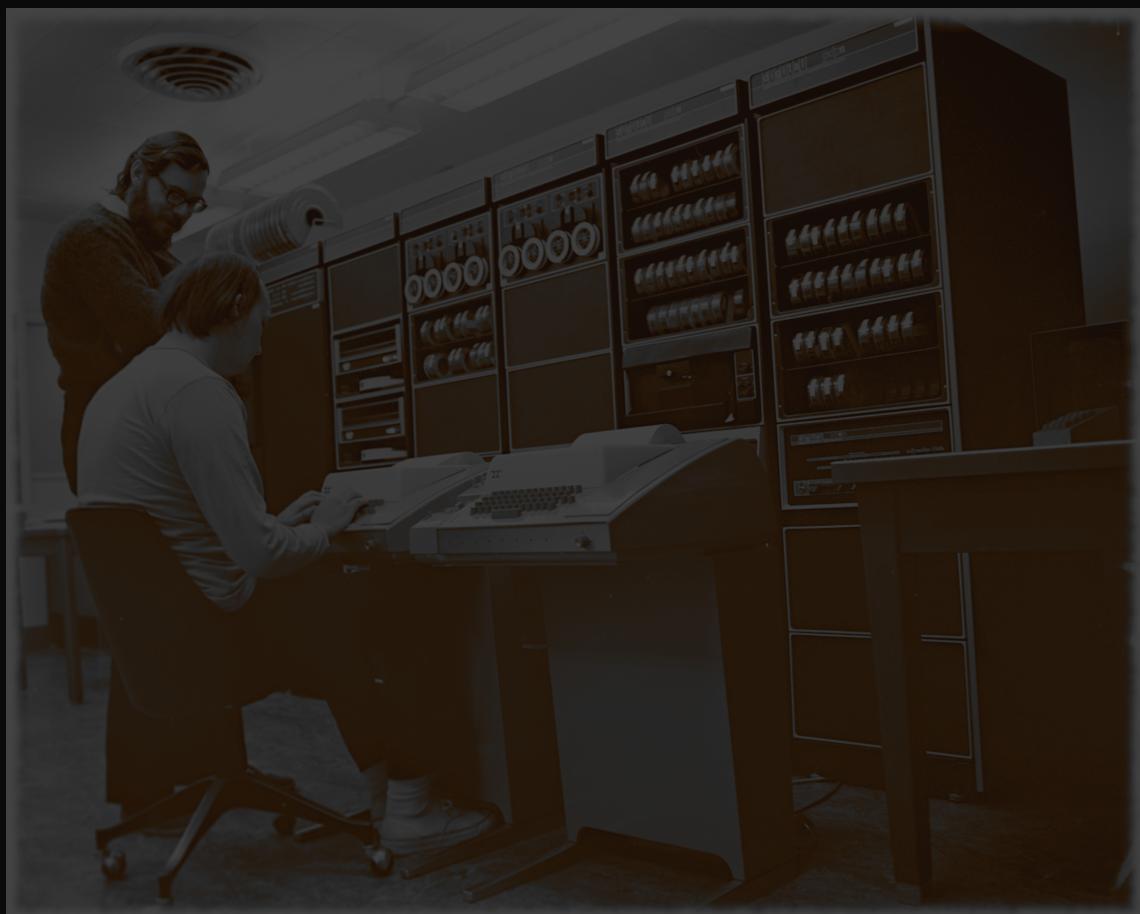


# Systèmes d'exploitation

Alain Lebret

2024-2025



## 💻 TP de criblage par processus filtrants – éléments de réponse

```
/*
 * ENSICAEN
 * 6 Boulevard Maréchal Juin
 * F-14050 Caen Cedex

 * Exemplier du cours de systèmes d'exploitation (OS)
 * TP sur les tubes : crible d'Eratosthenes à l'aide d'un pipeline
 *
 * Copyright (C) 1995-2024 Alain Lebret (alain.lebret (at) ensicaen.fr)
 */

#ifndef ERATOSTHENES_H
#define ERATOSTHENES_H

/***
 * @file eratosthenes_fork.h
 */
#define SUCCES 0
#define STOP -1
#define SORTIE 0
#define ENTREE 1

FILE *sortie; /* Fichier que tous les processus utilisent */

/***
 * Traitement des erreurs de création du tube anonyme.
 */
void traiter_erreur_pipe();

/***
 * Traitement des erreurs de clonage du processus, par exemple en cas
 * de table des processus pleine.
 */
void traiter_erreur_fork();

/***
 * Fonction récursive permettant de créer un nouveau noeud dans le pipeline.
 * Ne traite qu'un seul nombre et ses multiples. Le nombre premier suivant
 * ↴ est
 * passé au noeud suivant.
```

```
* @param tube le tube en entrée permettant au noeud de recevoir les nombres
* @param premier Premier nombre "premier" de la série reçue par le noeud.
*/
void passer_au_suivant(int tube[2], int premier);

/***
 * Ecriture par le père dans le premier tube créé entre lui et son fils
 * aîné, de tous les nombres compris entre 2 et max. Le dernier nombre
 * écrit est -1.
 *
 * @param tube Descripteurs du tube auquel le père est connecté à son
 *             fils aîné.
 * @param max Nombre jusqu'auquel on recherche les nombres premiers.
 */
void gerer_pere(int tube[2], int max);

/***
 * Lecture par le fils aîné dans le premier tube créé entre lui et son
 * père du premier nombre (c'est-à-dire 2) qui est "premier". Celui-ci
 * est alors écrit dans le fichier.
 * Le dernier nombre écrit doit être -1.
 *
 * @param tube Descripteurs du tube auquel le père est connecté à son
 *             fils aîné.
 * @param max Nombre jusqu'auquel on recherche les nombres premiers.
 */
void gerer_fils_aine(int tube[2]);

#endif

/*
 * ENSICAEN
 * 6 Boulevard Maréchal Juin
 * F-14050 Caen Cedex

 * Exemplier du cours de systèmes d'exploitation (OS)
 * TP sur les tubes : crible d'Eratosthenes à l'aide d'un pipeline
 *
 * Copyright (C) 1995-2024 Alain Lebret (alain.lebret (at) ensicaen.fr)
 */
```

```
/**  
 * @file eratosthenes_fork.c  
 */  
  
#include <sys/types.h>  
#include <sys/wait.h>  
#include <stdio.h>  
#include <unistd.h>  
#include <stdlib.h>  
#include "eratosthenes_fork.h"  
  
extern FILE *sortie;  
  
void traiter_erreur_pipe()  
{  
    perror("Création du tube impossible\n");  
    exit(EXIT_FAILURE);  
}  
  
void traiter_erreur_fork()  
{  
    perror("Création du fils impossible.\n");  
    exit(EXIT_FAILURE);  
}  
  
void passer_au_suivant(int tube[2], int premier)  
{  
    int val = 0; /* tous les entiers sont stockés dans val */  
    int nouveau_tube[2];  
    int etat = SUCCES;  
    int pid = 0;  
  
    while (val != STOP) {  
        read(tube[SORTIE], &val, sizeof(int));  
  
        if (val % premier != 0) {  
            if (val != STOP) {  
                if (pipe(nouveau_tube) == -1) {  
                    traiter_erreur_pipe();  
                }  
                switch (pid = fork()) {  
                    case -1:  
                        traiter_erreur_fork();  
                    case 0:  
                        close(tube[SORTIE]);  
                        close(tube[1]);  
                        close(nouveau_tube[0]);  
                        write(nouveau_tube[1], &val, sizeof(int));  
                        exit(0);  
                    default:  
                        close(tube[0]);  
                        close(nouveau_tube[1]);  
                        read(nouveau_tube[0], &val, sizeof(int));  
                }  
            }  
        }  
    }  
}
```

```
        case 0:
            close(nouveau_tube[ENTREE]);
            read(nouveau_tube[SORTIE], &val, sizeof(int));
            fprintf(sortie, "%d\n", val);
            /*
             * Père et fils partagent le même environnement.
             * Au moment de la fermeture par fclose(), chaque
             * fils enregistre TOUT ce qui a été enregistré
             * par son père :
             */
            fflush(sortie);
            passer_au_suivant(nouveau_tube, val);
        default:
            close(nouveau_tube[SORTIE]);
    }
}
write(nouveau_tube[ENTREE], &val, sizeof(int));
}
}
close(tube[SORTIE]);

if (pid != 0) {
    wait(&etat);
    close(nouveau_tube[ENTREE]);
}
fclose(sortie);
exit(etat);
}

void gerer_pere(int tube[2], int max)
{
    int val;

    close(tube[SORTIE]);

    for (val = 2; val <= max; val++)
        write(tube[ENTREE], &val, sizeof(int));

    /* Message de fin pour ses fils */
    val = STOP;
    write(tube[ENTREE], &val, sizeof(int));
}
```

```
void gerer_fils_aine(int tube[2])
{
    int val;

    close(tube[ENTREE]);
    read(tube[SORTIE], &val, sizeof(int));
    fprintf(sortie, "%d\n", val);
    fflush(sortie);
    passer_au_suivant(tube, val);
}

/*
 * ENSICAEN
 * 6 Boulevard Maréchal Juin
 * F-14050 Caen Cedex

 * Exemplier du cours de systèmes d'exploitation (OS)
 * TP sur les tubes : crible d'Eratosthenes à l'aide d'un pipeline
 *
 * Copyright (C) 1995-2024 Alain Lebret (alain.lebret (at) ensicaen.fr)
 */

/***
 * @file main.c
 */
#include <sys/types.h>
#include <sys/wait.h>
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
#include "eratosthenes_fork.h"

extern FILE *sortie; /* Fichier que tous les processus utilisent */

int main(int argc, char **argv)
{
    int tube[2];
    int etat;
    int max;
```

```
etat = SUCCES;
if (argc < 2) {
    max = 7;
} else {
    max = atoi(argv[1]);
}
sortie = fopen("nombrespremiers.txt", "w");

if (pipe(tube) == -1) {
    traiter_erreur_pipe();
}

switch (fork()) {
    case -1:
        traiter_erreur_fork();
        break;
    case 0: /* action du fils */
        gerer_fils_aine(tube);
        break;
    default: /* action du pere */
        gerer_pere(tube, max);
}
wait(&etat);
close(tube[ENTREE]);
fclose(sortie);

return etat;
}
```