```c
#include <errno.h>
#include <fcntl.h>
#include <signal.h>
#include <stdarg.h>
#include <stdint.h>
#include <stdio.h>
#include <stdlib.h>
#include <stdnoreturn.h>
#include <string.h>
#include <time.h>
#include <unistd.h>

#define CHEMIN_MAX 512
#define MAX_INT_LEN 20  // Le plus grand entier possède 20 caractères

// Vérifie les appels aux fonctions qui renvoient -1 en cas d'erreur
#define CHK_PS(v)          \
    do {                   \
        if ((v) == -1)     \
            raler(1, #v); \
    } while (0)

// Vérifie les appels à la finction 'signal'
#define CHK_SG(v)           \
    do {                    \
        if ((v) == SIG_ERR) \
            raler(1, #v);   \
    } while (0)

// Rale
noreturn void raler(int syserr, const char *fmt, ...) {
    va_list ap;
    va_start(ap, fmt);
    vfprintf(stderr, fmt, ap);
    fprintf(stderr, "\n");
    va_end(ap);
    if (syserr)
        perror("");
    exit(1);
}

// Valeurs atomiques pour gèrer les signaux
volatile sig_atomic_t sig_alrm, sig_int, sig_term;

void fct_alrm(int sig) {
    (void)sig;
    sig_alrm = 1;
}

void fct_int(int sig) {
    (void)sig;
    sig_int = 1;
}

void fct_term(int sig) {
    (void)sig;
    sig_term = 1;
}

int main(int argc, char *argv[]) {
    if (argc != 4)
        raler(0, "Nombre d'arguments incorrect");

    if (strlen(argv[1]) > CHEMIN_MAX)
        raler(0, "Chemin trop long");

```

Ah ? Vous en êtes sûr ?
=> non portable

Faites un messsage utile : rappelez la syntaxe


```c
    int tinc = atoi(argv[2]);
    int tstamp = atoi(argv[3]);
    if (tinc < 0 || tstamp <= 0)
        raler(0, "Temps invalide");

    alarm(tstamp);

    CHK_SG(signal(SIGALRM, fct_alrm));
    CHK_SG(signal(SIGINT, fct_int));
    CHK_SG(signal(SIGTERM, fct_term));

    int fd;
    CHK_PS(fd = open(argv[1], O_WRONLY | O_CREAT | O_TRUNC, 0666));

    time_t time_epoch;
    char *time_formated;

    char compteur_str[MAX_INT_LEN + 1];
    int write_size;

    sig_alrm = sig_int = sig_term = 0;
    for (uintmax_t compteur = 0;; compteur++) {
        if (usleep(tinc * 1000) == -1) {
            if (EINTR)
                fprintf(stderr, "\tSommeil interrompu\n");
            else if (EINVAL)
                raler(1, "'tinc' trop grand pour ce système");
        }

        if (sig_alrm) {
            if ((time_epoch = time(NULL)) == (time_t)-1)
                raler(1, "Erreur de la fonction 'time'");
            time_formated = ctime(&time_epoch);
            CHK_PS(write(fd, time_formated, 25));
            sig_alrm = 0;
            alarm(tstamp);
        }

        if (sig_int) {
            if ((write_size = snprintf(compteur_str, MAX_INT_LEN + 1, "%ju%c",
                                       compteur, '\n')) < 0)
                raler(0, "Erreur de conversion du compteur en chaîne");
            CHK_PS(write(fd, compteur_str, write_size));
            sig_int = 0;
        }

        if (sig_term) {
            CHK_PS(write(fd, "fin\n", 4));
            CHK_PS(close(fd));
            return 0;  // Indique que le programme s'est arété comme on voulait
        }
    }
}
```

L'énoncé autorise les fcts de bib (ex : fopen/fprintf) : ce serait plus simple

Le nom "epoch" prête à confusion

dans ce cas là, il ne faudrait rien faire (rien afficher)