SIGINT -> read long string returns NULL
(ERINT) -> get back
-> exec does not inherit Sig handler
-> waiting for children (ERINT is set)
adjust amount of children to wait for.

POSIX reliable signal handling:

↳ sigaction (2)
↳ changes the action in response to a signal

↳ procmask
↳ feature of a process
↳ set of signals to be held pending

↳ waiting:
int pause(void)
int sigsuspend(sigset_t *mask);

signal sets (sigsetops)

sigset_t

```
int sigemptyset(sigset_t *set);
int sigfillset(sigset_t *set);
int sigaddset(sigset_t *set, int sig)
int sigdelset(sigset_t *set, int sig)
int sigismember(sigset_t *set, int sig)
```

```
int sigprocmask(int how, sigset_t *set, sigset_t *old)
```

how: • SIG_BLOCK
• SIG_UNBLOCK
• SIG_SETMASK

int sigaction(int sig, struct sigaction *sa, struct sigaction *old);

struct sigaction {

    void (*sa_handler)(int);
    void (*sa_sigaction)(int, siginfo_t *, void *);   } only define one

    sigset_t sa_mask;
    int   sa_flags; → SA_RESTART
                           SA_NODEFER
                           SA_RESETHAND
                           SA_SIGINFO

}

```
void handler(int num){
    printf("Neener neener!\n");
}

int main(int argc, char *argv[]){
    int num;
    struct sigaction sa;
    sigset_t set, old;

    sa.sa_handler = handler;
    sigemptyset(&sa.sa_mask);
    sa.sa_flags = 0;

    sigemptyset(&set);
    sigaddset(&set, SIGINT);
    sigprocmask(SIG_BLOCK, &set, &old);

    sigaction(SIGINT, &sa, NULL);   <- error checking
    sigdelset(&old, SIGINT);

    for(num=0; num<3; num++){
        sigsuspend(&old);
    }

    return 0;
}
```

or

```
sigprocmask(SIG_UNBLOCK, &set, NULL);
pause();
```