# Quelques codes pour les signaux

## 1) Code minimal envoie / réception d'un signal

```
void signal_handler(int n_sig){
    /*
     * Code gestionnaire de signal
     * Souvent un switch(...){...}
     */
    return;
}

int main(){
    printf("PID %d\n", getpid());

    if(signal(..., signal_handler) == SIG_ERR){
        // gestion d'erreur de réception
        perror("message");
        return;
    }
    //envoyer un signal au [PID]
    kill([PID], [SIG]);

    while(1) pause();
    return 0; //code non atteignable
}
```

Ce code permet de gérer la réception du signal [SIG] au processus [PID].
`while(1) pause();` permet au programme de rester en vie et de gérer d'autres signaux.

## 2) Masquer / démasquer un signal

On admet la présence de la fonction signal_handler au même titre que l'exemple précédent.

```
int main (){
    // permettre la réception de [SIG]
    if (signal([SIG], signal_handler) == SIG_ERR){
        // traitement si erreur de réception
    }
    // bloquer un signal
    sigset_t mask, pend;
    sigemptyset (&mask);              // init ensemble
    sigadset (&mask, [SIG]);          // ajout d'un signal
    sigprocmask(SIG_BLOCK, &mask, NULL); // bloquer les signaux

    // démasquage d'un signal
    sigdelset (&mask, [SIG]);

    return 0;
}
```

étapes pour bloquer un signal

| | |
|---|---|
| INIT | initialiser un ensemble de signaux |
| ADD | ajouter le/les signaux à l'ensemble |
| BLOCK | bloque l'ensemble des signaux |

## 3) La structure sigaction

- Création sigaction
- bloquage d'un signal [SIG]

```
int main (){
    :
    // création & initialisation sigaction
    struct sigaction sa;
    sa.sa_handler = signal_handler;
    sa.sa_flags = 0;
    sigemptyset (&sa.sa_mask);
    sigaddset (&sa.sa_mask, [PID]);
    sigprocmask (SIG_BLOCK, &sa.sa_mask, NULL);
    sigaction ([SIG], &sa, NULL);   // réception
    // envoyer un signal
    kill ([PID], [SIG]);
    :
    return 0;
}
```

bloquage d'un signal

La structure struct sigaction

| | |
|---|---|
| sa_handler | gestionnaire de signaux |
| sa_flags | flags gestion signal |
| sa_mask | ensemble sigset_t de signaux masqués |

⚠ important de vider sa_mask au départ car il peut ne pas être vide.

## 4) Pseudo démons

↳ le programme se détache du terminal et s'exécute en arrière plan

Code minimal : alarme qui se répète un laps de temps. n

```
void alarm_handler(int n_sig){
    /*
     * gestionnaire de l'alarme
     */
    alarm(n); // reprogrammer l'alarme
}

int main(){
    signal(SIGALRM, alarm_handler);
    alarm(n); // 1ère programmation de l'alarme

    while(1) pause();
}
```

version signal

```
int main(){
    struct sigaction sa = {0};
    sa.sa_handler = alarm_handler
    sigaction(SIGALRM, &sa, NULL);
    alarm(n);

    while(1) pause();
}
```

version sigaction

# Examen 2024

```
void alarm_handler(int n_sig){
    switch(n_sig){
        case SIGALRM:
            if(upower() < 5){
                system("mail -s <<batterie faible>>
                        admin@univ-lehavre.fr <<votre
                        batterie est faible>>");
                alarm(0); // déprog l'alarme
            }
    }
    alarm(240); //4min = 60s
}

int main(){
    struct sigaction sa = {0};
    sa.sa_handler = alarm_handler;
    sigaction(SIGALRM, &sa, NULL);
    alarm(240);      // prog alarme

    while(1) pause();
}
```