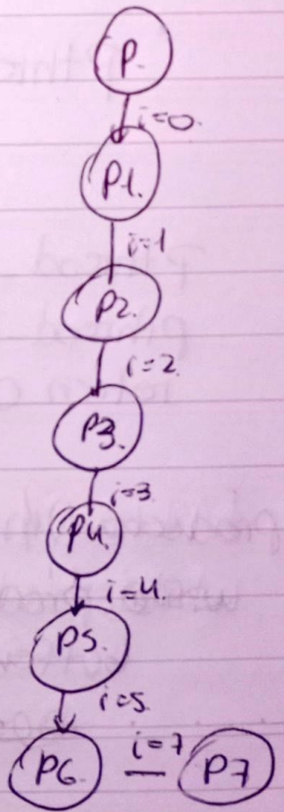
# Recu (2023)

Dado el sg fragmento de código de ej1.c que se ejecuta como $./ej1 8.

```
Void sig_usr2(int);
  pid_t pid2;

int main (int argc, char **argv){
      int i;
      pid2 = getpid()
      pid_t pid;
      signal(SIGUSR2, sig_usr2);  → 8.
       for(i=0; i<atoi(argv[1]); i++){
            pid = fork();
            If (pid > 0) break;
      }
→ //.    If (pid > 0){ while (wait(NULL) > 0); } else {

      Printf("soy %d hijo de %d \n", getpid(), getppid());
      Pause();
      return 0;
}  }

Void sigusr2 (int s) {
→ //.  kill (pid, SIGTERM);
       exit(0); (proceso ppal).

}
```

Completar pl que todo finalice



③ los Hilos prod y consumidor se ejecutan durante 10 seg.
(prod y cons letras del abecedario)

escribir el código de f-Alarma() y consumidor().

```
#define BUFFER_SIZE 10

void *productor();
void *consumidor();
void f_alarma(int); ⟶ valor de salir.

char buffer[BUFFER_SIZE];
int salir = 0;
int n = 0;
pthread_mutex_t sc = PTHREAD_MUTEX_INITIALIZER;

int main(int argc, char **argv){
    signal(SIGALARM, f_alarma);
    alarm(10);
    pthread_t p, c;

    pthread_create(&p, NULL, productor, NULL);
                  (&c, NULL, consumidor, );

    pthread_join(p, NULL);
    pthread_join(c, NULL);
    return 0;

void f_alarma(int s){
    salir = 1;
}
```

```
void *Productor (){ -> !salir.
    while(){
        Pthread_mutex_lock(&sc);
        if (cant < BUFFER_SIZE){
            buffer[cant] = 'x'; -> Producir();
            n++;
        }
        pthread_mutex_unlock(&sc);
    }
}

void *consumidor (){ -> !salir
    while(1){
        Pthread_mutex_lock(&sc);
        if( count > 0 ){
            printf("consumi: %c\n", buffer[count]);
            cant--;
        }
        pthread_mutex_unlock(&sd);
    }
}
```