PARCIAL 16/05/2022

① Dado el sig. fragmento de código

```
vad f_child(int)
vad f_alarma(int);
int salir=0;
int main(int'arge, char *argv[]){
  pid_t pid;
  Pid=fork();
  if(pid==0){
      signal(SIGALRM, f_alarma);
      alarm(2);
      Printf("HOLA MUNDO!/n");
       while(!salir);
      }else{
         signal(SIGCHLD, f_child);
         while(!salir);
      }
    return 0;
}
void f_child(ints){ printf("f_child"); wait(0); salir=1;}
vad f_alarm(ints){ printf("f_alarma():/n"); salir=1;}
```

② ¿Qué muestra por pantalla?

HOLA MUNDO!
f_alarma():
f_child():

③ ¿El proceso genera procesos zombies?
No porque el padre hace wait(0) y recoge la info del hijo

(c) el proceso genera hijos huerfanos?
si tenemos la señal sigchild el proceso padre nunca termina antes que el hijo, así que no serán huerfanos

(d) si qutamos alarm(2) el hijo queda en un bop infinito ya que nunca se cambia la var. salir.

(E) si qutamos salir=1 de f-child el padre queda en un bop infinito

(4) Escribir un proceso que cree 5 hilos cuando se pulse CTRL-C por teclado
TAREA DE LOS HILOS INCREMENTAS UNA VAR GLOBAL mostrar el valor al final

```
void sigint(void);
void Hilo1(void);
int main(void){
    pthread_t h1;
    for(int i=0; i<5; i++){
        SIGNAL(SIG_INT, sigint);
        while(1){
            Pause();
        }
    }
return 0;
}

void Hilo1(void){
    varglobal ++;
    Printf("el valor de la variable es: %d /n", varglobal);
}

void sigint(void){
    Pthread_create = (&h1, NULL, (void*) Hilo1, NULL);
    Pthread_join(h1, NULL);
```

→ VAR GLOBAL = 0;

→ Printf("Presionar control-pl crear hilo");

PARCIAL 2022 (new)

A C ▷ (A○B) C ▷ (A○B) C ▷.

PA	PB	PC	PD
w(SA)	w(SAB)	w(SC)	w(SD)
sc	sc	sc	sc
s(sc)	s(sc)	s(SD)	(SAB)
w(SAB)			
sc			
s(sc)			

SA - SB - SC - SD - SAB
1 0 0 0 0

Como son semaforos binarios utilizo mutex.

```
void HiloA(void);
void HiloB(void);
void HiloC(void);
void HiloD(void);
```

```
pthread_mutex_t sa = pthread_inicializer;
pthread_mutex_t sc = pthread_inicializer;
pthread_mutex_t sd = pthread_inicializer;
pthread_mutex_t sab = pthread_inicializer;
```

```
int main (){
    Pthread h1;
    Pthread h2;
    pthread h3;
    pthread h4;

Pthread_mutex_lock (&sc);
pthread_mutex_lock (&sd);
Pthread_mutex_lock(&sab);

pthread_create(&h1, NULL, (void*)HiloA, NULL);
pthread_create(&h2, NULL, (void*)HiloB, NULL);
pthread_create (&h3, NULL, (void*)HiloC, NULL);
pthread_create (&h4, NULL, (void*)HiloD, NULL);
```

```
    pthread_join (h1, NULL); return 0;
}

void HiloA (void) {
    pthread_mutex_lock (sa);
    printf ("A \n");
    pthread_mutex_unlock (sc);
    pthread_mutex_lock (sab);
    printf ("A \n");
    pthread_mutex_unlock (sc);
}

void HiloB (void) {
    pthread_mutex_lock (sab);
    printf ("B \n");
    pthread_mutex_unlock (sc);
}

void HiloC (void) {
    pthread_mutex_lock (sc);
    printf ("C \n");
    pthread_mutex_unlock (sd);
}

void HiloD (void) {
    pthread_mutex_lock (sd);
    printf ("D \n");
    pthread_mutex_unlock (sab);
}
```