PARCIAL 1

(1) Realizar un programa que permita ingresar un comando por teclado y ejecute dicho comando en foreground. El ciclo debe terminar cuando el usuario ingresa "exit"

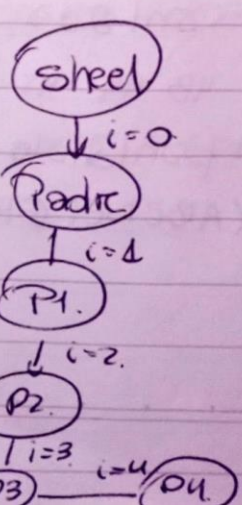
```
int main (int argc, char **argv){

    comando[256];

while(1){
    Printf("Ingrese un comando");
    scanf("%s", comando);
  If(strcmp(comando,"exit")==0){
    Printf("el programa finalizo");
    return 0;
  }else{
Int status = execlp(comando, comando, *char(NULL));
  }
  If(status==-1){
    Printf("error execlp");
  }
}
return 0;
}
```

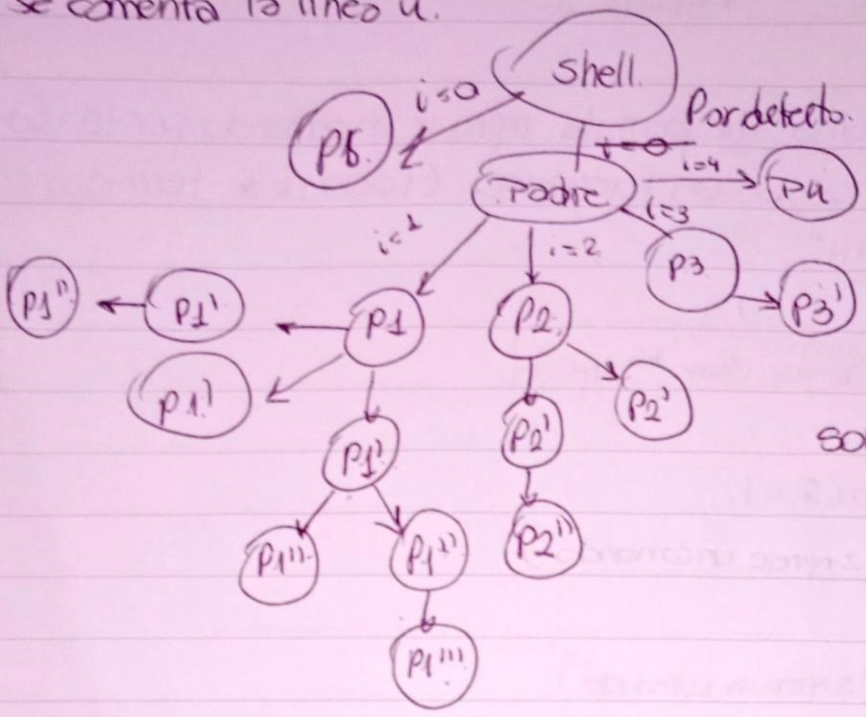
(2) Dado el sig fragmento de código

```
Int i;
for(i=0; i<5; i++){
  Pid_t pid = fork();
  If(pid>0) break
}
```



¿Cuántos procesos son creados? sin contar el shell 5.

(c) Si se comenta la linea 4.



son ~~16~~ 32 procesos sin contar el shell.

(3) Assembler.

(1) mov cx, 000A → muevo el contenido al registro cx

mov SI; 2000 → Inicializo el puntero en 2000.

mov DL; [SI] → muevo el contenido de ~~DS~~ SI al registro DL.

mov AH; 02 → muevo la función 02 (imprimir por pantalla) a la parte alta del registro Ax

Int 21 → Interrupción 21 ejecuta la función 02 que muestra el contenido de DL

Inc SI → Incremento el puntero SI

Dec CX → decremento ~~CS~~ CX

Jnz 0106 → si cx≠0 ejecuto nuevamente la instrucción (1)

Int 20 → si cx=0 finalizo el programa

41 42 43 44 45 46 47 48-49 50 00 00 00 00 00 00

muestra en pantalla 10 posiciones luego de la posición 2000.

Muestra (ABCDEFGHIJ)

Teoría

(A) ¿PARA QUE SIRVEN LAS FUNCIONES SIGSETJMP() y SIGLONGJMP()?

- Se usan para hacer saltos en el código

- sigsetjmp() permite determinar la parte del código donde se realizará el salto.

- siglongjmp() realiza el salto según lo indicado en sigsetjmp()

(B) ENUMERE 4 ORACIONES QUE DEFINAN QUÉ ES UN PROCESO:

(1) Programa en ejecución

(2) Instancia de un programa que se ejecuta en un computador

(3) Entidad que se le da uso del procesador.

(4) Unidad que presenta un unico hilo de ejecución, estado actual y recursos asignados.

(C) Objetivos del SO:

(1) FACILIDAD DE USO.

(2) Eficiencia.

(3) CAPACIDAD DE EVOLUCIÓN

(D) ¿Qué es un proceso zombie?

Es un proceso que finalizó pero no se le envió la información de este al proceso Padre

(E) DIF. ENTRE HILO / PROCESO

Los Hilos se crean mediante un proceso existente y se ejecutan de manera concurrente; comparten el mismo esp de memoria

Los procesos se crean mediante la llamada fork(), pueden iniciar y finalizar de manera independiente.