1: Realice la gráfica de la estructura de procesos generada por el siguiente código, en la gráfica asigne PID a cada proceso. (2)

```
int main(void){
      int n = 4;pid_t pid;
      while( n != 0){ pid = fork();
         if ( pid == 0 ) break;
         n--;
      } return n;
}
```

2: Modifique el siguiente código para que no exista ningún proceso huérfano ni zombie. (2.5)

```
int main(void){
   pid_t pid; pid = fork();
   if ( pid > 0 ){
      pid = fork();
      if ( pid > 0 ){
         pid = fork();
         if ( pid == 0 ) pid = fork();
      }
   } return 0;
}
```

3: Complete el siguiente programa para que el padre repita en pantalla "Padre dice hola" durante 3 segundos y luego terminen el padre y el hijo. (Usar señales, no usar sleep). (2.5)

```
void  mensaje(void);
int main(void) {
   pid_t pid;pid = fork();
   if (pid == 0)    { …… }
   else    { …… }
   return 0 ;
}
void mensaje(void) {
   printf("Padre dice hola\n");
}
```

4: Escriba en pseudocódigo la siguiente sincronización CABACABACABA ….. de ejecución de la sección crítica entre procesos, nombre los semáforos, indique valor inicial y escriba la sección de entrada y la sección de salida de cada proceso. (3)

Criterio de Aprobación: obtener -al menos- 6 puntos sobre un total de 10.

Teoría:

1. En un sistema de procesamiento batch o por lotes. Las instrucciones privilegiadas incluyen también operaciones de I/O, entonces, ¿Cómo es posible que un programa de usuario pueda realizar operaciones de I/O? (2)
2. Enumere 4 oraciones que definan qué es un proceso (2)
3. ¿Qué diferencia hay entre un hilo (thread) y un proceso? (2)
4. ¿Qué diferencia hay entre "trap" e interrupción? (2)
5. En un entorno Unix, ¿Qué es un proceso zombie? (2)

Criterio de Aprobación: obtener -al menos- 6 puntos sobre un total de 10.

Criterio de Calificación:
6 p = 6 (acorde a exactitud) 7 p = 7 8 p = 8 9 p = 9 10 p = 10 (acorde a exactitud)

Practica

1: Realice la gráfica de la estructura de procesos generada por el siguiente código, en la gráfica asigne PID a cada proceso. (2)

```
int main(void){
      int n = 4;pid_t pid;
      while( n != 0){ pid = fork();
         if ( pid == 0 ) break;
         n--;
      } return n;
}
```

1714 shell
  1991
    1992
    1993
    1994
    1995

2: Modifique el siguiente código para que no exista ningún proceso huérfano ni zombie. (2.5)

```
int main(void) {
     pid_t pid; pid = fork();
     if ( pid > 0 ){
          wait(0);
          pid = fork();
          if ( pid > 0 ){
               wait(0);
               pid = fork();
               if ( pid == 0 ) {
                    pid = fork();
                    if ( pid > 0 ) wait(0);
               } else {
                    wait(0);
               }
          }
     }
     return 0;
}
```

3: Complete el siguiente programa para que el padre repita en pantalla “Padre dice hola” durante 3 segundos y luego terminen el padre y el hijo. (Usar señales, no usar sleep). (2.5)

```
void handler(void);
void handler2(int);
int loop=1;
int main(int  argc, char **argv) {
        pid_t pid;
        pid = fork();
        if (pid == 0) {
                signal(SIGALRM,handler2);
                alarm(3);
                pause();
                kill(getppid(),SIGUSR1);
        } else {
                signal(SIGUSR1,handler2);
                while(loop)  handler();
                wait(0);
        }
        return 0 ;
}

void handler(void)  {
        printf("Padre  dice hola\n");
}

void handler2(int  signo)  {
        loop=0;
}
```

```
void handler(void);
void handler2(int);
int loop=1;
int main(int argc, char **argv) {
     pid_t pid;
     pid = fork();
     if (pid == 0) {
          alarm(3);
          pause();  // while(loop);
     } else {
          signal(SIGALRM,handler2);
          alarm(3);
          while(loop) handler();
          wait(0);
     }
     return 0 ;
}

void handler(void) {
```

```
    printf("Padre dice hola\n");
}

void handler2(int signo) {
    loop=0;
}
```

4: Escriba en pseudocódigo la siguiente sincronización CABACABACABA ….. de ejecución de la sección crítica entre procesos, nombre los semáforos, indique valor inicial y escriba la sección de entrada y la sección de salida de cada proceso. (3)

Sa=0 Sb=0 Sc=1		
A	B	C
wait(Sa) SC signal(Sb) wait(Sa) SC signal(Sc)	wait(Sb) SC signal(Sa)	wait(Sc) SC signal(Sa)

O bien:

Sa = 0 Sb = 0 Sc = 1 Sx = 1		
A	B	C
wait(Sa) SC Signal(Sx)	wait(Sb) wait(Sx) SC Signal(Sc) Signal(Sa)	wait(Sc) wait(Sx) SC Signal(Sb) Signal(Sa)

Teoría

1 En un sistema de procesamiento batch o por lotes, Las instrucciones privilegiadas incluyen también operaciones de I/O, entonces, ¿Cómo es posible que un programa de usuario pueda realizar operaciones de I/O?

Porque las hace a través del monitor y no directamente

2 Enumere 4 oraciones que definan qué es un proceso

Programa en ejecución.

Instancia de un programa ejecutado en un computador.
"Espíritu animado" de un programa.
Entidad asignada a una CPU y ejecutada por ésta.
Unidad de actividad que se caracteriza por la ejecución de una secuencia de instrucciones, un estado actual y un conjunto de recursos del sistema asociados.

3 ¿Qué diferencia hay entre un hilo (thread) y un proceso?
Un proceso es una unidad de asignación de recursos y de protección, la cual puede ser implementada bajo un modelo de un único hilo o bajo un modelo multihilo.
Cada hilo tiene su información contextual (BCP), su pila o stack, pero comparte el BCP del proceso al que pertenece el hilo y tiene acceso al espacio de direcciones del proceso común a todos los hilos, ver Fig. 4.2 Pag. 160.
Cada hilo tiene: su estado de ejecución, contexto, pila, espacio de direcciones para variables locales, acceso a memoria y recursos del proceso, compartido por todos los hilos de su mismo proceso.

4 ¿Qué diferencia hay entre "trap" e interrupción?
Son conceptos similares, pero no no idénticos, un trap es una forma de interrupción pero que se origina dentro del propio proceso interrumpido, asociado a una condicion de error o excepción. Mientras que una interrupción es algo externo al proceso en ejecución, por ejemplo, puede ser que se haya agotado la rodaja de tiempo asignada para la ejecucion del mismo (time slice) entonces se produce una interrupción de reloj (clock).

5 En un entorno Unix, ¿Qué es un proceso zombie?
Es un proceso que ya no esta en ejecución, ya sea por una terminación normal o anormal o porque fue abortado; pero aún queda su información registrada en el sistema para que pueda consultarla el padre. Obsérvese el código del shell, cuando el proceso padre hace un fork() (crea un hijo), y luego hace una llamada a la función waitpid() (si el hijo termina antes de la llamada a waitpid() por parte del padre, el hijo pasa a estado zombie) y luego el padre necesita que no "desaparezca su rastro" para recuperar el status de finalización del proceso hijo. Una vez que el padre ha recuperado la información de finalización del proceso hijo a través de waitpid(), el hijo deja de estar en estado zombie y pasa al estado terminado [Tanenbaum, Sistemas Operativos Modernos, 3ra. ed., Pag. 744].