PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ
FACULTAD DE CIENCIAS E INGENIERÍA

SISTEMAS OPERATIVOS

1ra práctica (tipo a)
(Segundo semestre de 2016)

Horario 0781: prof. V. Khlebnikov
Horario 0782: prof. F. Solari A.

Duración: 1 h. 50 min.
Nota: No se puede usar ningún material de consulta.
La presentación, la ortografía y la gramática influirán en la calificación.
Puntaje total: 20 puntos

Pregunta 1 (4 puntos – 20 min.) Responda las siguientes cuestiones en forma breve, pero completa:

(a) (2 puntos – 10 min.) Considere una máquina expendedora de gaseosas. El microcontrolador o microcomputador tiene como dispositivos de E/S un teclado, una pantalla, acciona el mecanismo de expendio y recibe señales de éste y del receptáculo de monedas. Sin embargo, ejecuta permanentemente el mismo programa, pudiendo activarse con otros interruptores internos las funciones de emisión de reportes y otras como conteo de monedas durante el modo de servicio.

¿Es razonable contar con un sistema operativo para ese microcontrolador o microcomputador especializado? En caso su respuesta sea afirmativa, ¿cómo aplicaría las definiciones de sistema operativo para el caso? En caso su respuesta sea negativa, ¿por qué no podrían aplicarse las definiciones?

(b) (2 puntos – 10 min.) El modelo de memoria de un proceso en general, tiene partes, áreas o segmentos definidos, ¿con qué nombre se les conoce? ¿Estas partes deben ser contiguas entre ellas o debe haber contigüidad en su contenido? Justifique la contigüidad.

Pregunta 2 (6 puntos – 30 min., cada pregunta es de 1 punto)

(a) If the waiting time for a process is time fraction p and there are n processes in the memory, ¿what is p^n?

(b) Suppose a new process in a system arrives at an average of six processes per minute, and each process requires an average of 8 seconds of service time. Estimate the fraction time the CPU is busy in a system with a single processor.

(c) The state of a process after it encounters an I/O instruction is … (complete).

(d) A processing mode with the execution of a series of programs each on a set of inputs, rather than a single input, without manual intervention (non-interactive) is called … (complete).

(e) Documents formatted for printing are stored in a queue at the speed of the computer, then retrieved and printed at the speed of the printer. This is an example of the most common use of the technique called … (complete).

(f) ¿Qué significan los diferentes sufijos en *execl*, *execv*, *execle*, *execve*, *execlp*, *execvp*?

Pregunta 3 (5 puntos – 25 min.) Analice el siguiente código:

```
$ cat -n 2016-2_pr1.c
     1  #include <sys/wait.h>
     2  #include <stdio.h>
     3  #include <stdlib.h>
     4  #include <unistd.h>
     5
     6  int
     7  main(int argc, char **argv)
     8  {
     9    pid_t pid, pid1=getpid();
    10    int n=++argc, last=1;     /* argc se queda modificada */
    11
    12    while ((pid=fork()) || (pid1==getppid())) {
    13      if (!pid && !n && last) {
    14        last--; n=argc;       /* argc modificada */
    15      }
    16      if (!n) break;
```

```
    17      n--;
    18    }
    19    while(wait(NULL) != -1);
    20    printf("Exiting: pid=%d, ppid=%d\n", getpid(), getppid());
    21    exit(0);
    22  }
```

```
$ gcc 2016-2_pr1.c -o 2016-2_pr1
$ ./2016-2_pr1
Exiting: pid=...
```

En la forma de ejecución presentada, describe la relación entre los procesos creados.

Pregunta 4 (5 puntos – 25 min.) (*Keith Haviland, Ben Salama*) Analice el siguiente código en lenguaje C:

```
$ cat -n testjoin.c
     1  #include <stdio.h>
     2  #include <stdlib.h>
     3  #include <sys/types.h>
     4  #include <sys/wait.h>
     5  #include <unistd.h>
     6
     7  fatal(s) /* print error message and die */
     8  char *s;
     9  {
    10     perror(s);
    11     exit(1);
    12  }
    13
    14  int join(com1, com2)
    15  char *com1[],*com2[];
    16  {
    17     int p[2], status;
    18
    19          /* create .....................*/
    20          switch(fork()) {
    21                  case -1:
    22                          fatal("1st fork call in join");
    23                  case  0:
    24                          break;
    25                  default:
    26                          wait(&status);
    27                          return(status);
    28          }
    29          /* remainder .................*/
    30
    31          /* make ......................*/
    32          if(pipe(p) < 0)
    33                  fatal("pipe call in join");
    34
    35          /*create .....................*/
    36          switch(fork()) {
    37                  case -1:                /*.............*/
    38                          fatal("2nd fork call in join");
    39                  case 0:
    40                          close(1);       /*.............*/
    41                          dup(p[1]);      /*.............*/
    42
    43                          close(p[1]);/*.............*/
    44                          close(p[0]);
    45
    46                          execvp(com1[0],com1);
    47                          /*.............*/
    48                          fatal("1st execvp call in join");
    49                  default:
    50                          close(0);  /*.............*/
    51                          dup(p[0]); /*.............*/
    52
    53                          close(p[0]); /*.............*/
    54                          close(p[1]);
    55
    56                          execvp(com2[0],com2);
    57                          /*.............*/
    58                          fatal("2nd execvp call in join");
    59          }
    60  }
    61
    62  int main()
    63  {
    64
```

```
65      char *one[4], *two[3];
66      int ret;
67           /*............*/
68      one[0]="ls";
69      one[1]="-l";
70      one[2]="/usr/lib";
71      one[3]=NULL;
72      /*............*/
73           two[0]="grep";
74      two[1]="java";
75      two[2]=(char *)0;
76           /*............*/
77      ret=join(one,two);
78      printf("join returned %d\n",ret);
79      exit(0);
80  }
```

(a) (2 puntos – 10 min.) Explique cuántos procesos se tienen durante la ejecución de **`testjoin`**, el funcionamiento de cada proceso y cómo se relacionan (haga un esquema si lo cree necesario).

(b) (1 punto – 5 min.) Agregue los comentarios de la función `main` (líneas 62 a 80).

(c) (2 puntos – 10 min.) Agregue los comentarios de la función `join` (líneas 14 a 60).

La práctica ha sido preparada por FS (1,4) y VK(2,3)
con LibreOffice Writer en Linux Mint 18 Sarah.

Profesores del curso: (0781) V. Khlebnikov
(0782) F. Solari A.

Pando, 9 de septiembre de 2016