PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ
FACULTAD DE CIENCIAS E INGENIERÍA

SISTEMAS OPERATIVOS

1ra práctica (tipo a)
(Segundo semestre de 2015)

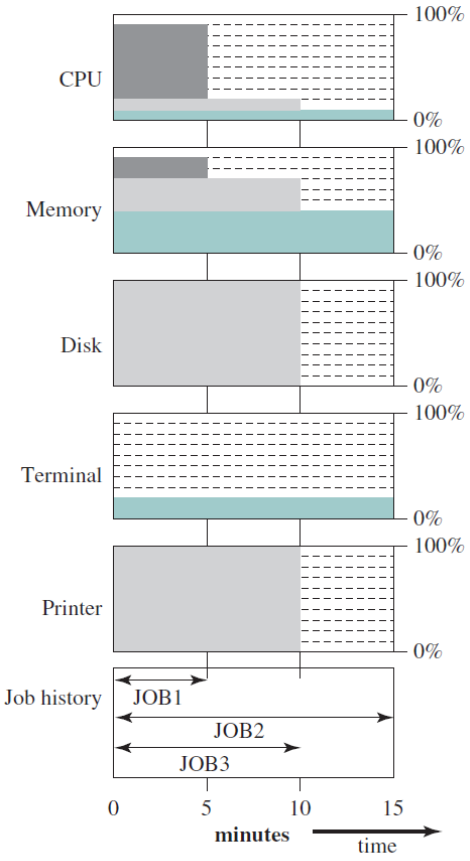
Horario 0781: prof. V. Khlebnikov

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 (2 puntos – 10 min.) (*OSIDP8E, Chapter 2*) Justifique los valores de utilización de recursos para el caso de multiprogramación presentados en la Tabla 2.2 usando el histograma también presentado.

Table 2.2 Effects of Multiprogramming on Resource Utilization

	Uniprogramming	Multiprogramming
Processor use	20%	40%
Memory use	33%	67%
Disk use	33%	67%
Printer use	33%	67%
Elapsed time	30 min	15 min
Throughput	6 jobs/hr	12 jobs/hr
Mean response time	18 min	10 min



(b) Multiprogramming

Pregunta 2 (3 puntos – 15 min.) (*OSIDP8E, Chapter 3*) Assume that at time 5 no system resources are being used except the processor and memory. Now consider the following events:

At time 5: P1 executes a command to read from disk unit 3.
At time 15: P5's time slice expires.
At time 18: P7 executes a command to write to disk unit 3.
At time 20: P3 executes a command to read from disk unit 2.
At time 24: P5 executes a command to write to disk unit 3.
At time 28: P5 is swapped out (from memory to disk).
At time 33: An interrupt occurs from disk unit 2; P3's read is complete.
At time 36: An interrupt occurs from disk unit 3; P1's read is complete.
At time 38: P8 terminates.
At time 40: An interrupt occurs from disk unit 3; P5's write is complete.
At time 44: P5 is swapped back in (from disk to memory).
At time 48: An interrupt occurs from disk unit 3; P7's write is complete.

For each time 22, 37, and 47, identify which state each process is in.

Pregunta 3 (4 puntos – 20 min.) Complete la salida producida durante la ejecución del siguiente programa. ¡No olvide indicar TODOS los procesos que formarán el árbol! Es que es fácil olvidar algunos.

```
$ cat 2015-2_LSO-1_3A_AB.c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <signal.h>
#include <sys/types.h>

int
main(int n, char *argv[])
{
    int i,j,np;
```

```
    char str[20];

    if (n != 2) {
        printf("Usage: %s <base>\n", argv[0]);
        exit(1);
    }
    np = atoi(argv[1]);
    for (i=0; i<np; i++)
        if (!fork()) {
            for (j=0; j<i; j++)
                if (fork()) break;
            pause(); /* sleep until a signal is delivered */
        }
    sleep(1);
    sprintf(str,"pstree -lp %d", getpid());
    system(str);
    kill(0,SIGKILL); /* kill all process group */
    exit(0);
}
$ gcc 2015-2_LSO-1_3A_AB.c -o 2015-2_LSO-1_3A_AB
$ ln 2015-2_LSO-1_3A_AB 3A
$ ./3A 3
3A(2852)─┬─...
         ├─...
...
Terminado (killed)
```

Pregunta 4 (3 puntos – 15 min.) Modifiquemos un poco el programa anterior. ¿Cómo ahora será el árbol de procesos presentado en la salida?

```
$ cat 2015-2_LSO-1_3B_AB_VK.c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <signal.h>
#include <sys/types.h>

int
main(int n, char *argv[])
{
    int i,j,base,half_base,k;
    char str[20];

    if (n != 2) {
        printf("Usage: %s <base>\n", argv[0]);
        exit(1);
    }

    base = atoi(argv[1]); half_base = base>>1;
    for (k=0,i=0; k<base; k<half_base?i++:i--,k++)
        if (!fork()) {
            for (j=0; j<i; j++)
                if (fork()) break;
            pause();
        }
    sleep(1);
    sprintf(str,"pstree -lp %d", getpid());
    system(str);
    kill(0,SIGKILL);
    exit(0);
}
$ gcc 2015-2_LSO-1_3B_AB_VK.c -o 2015-2_LSO-1_3B_AB_VK
$ ln 2015-2_LSO-1_3B_AB_VK 3B
$ ./3B 7
3B(2936)─┬─...
         ├─...
...
Terminado (killed)
```

Pregunta 5 (2 puntos – 10 min.) La siguiente versión del mismo programa tiene cambios en las últimas líneas de código. Explique qué resultados provocará la ejecución de este programa. También indique los cambios en el árbol de procesos.

```
$ cat 2015-2_LSO-1_3C_AB_VK.c
#include <stdio.h>
...
        }
    sleep(1);
    sprintf(str,"%d",getpid());
    execl("/usr/bin/pstree","pstree","-lp",str,NULL);
```

```
    kill(0,SIGKILL);
    exit(0);
}
$ gcc 2015-2_LSO-1_3C_AB_VK.c -o 2015-2_LSO-1_3C_AB_VK
$ ln 2015-2_LSO-1_3C_AB_VK 3C
$ ./3C 7
...
```

Pregunta 6 **(2 puntos – 10 min.)** La última versión del mismo programa tiene los siguientes cambios. Explique qué resultados provocará la ejecución de este programa.

```
$ cat 2015-2_LSO-1_3D_AB_VK.c
#include <stdio.h>
...
    for (k=0,i=0; k<base; k<half_base?i++:i--,k++)
        if (!fork()) {
            for (j=0; j<i; j++)
                if (fork()) break;
            k?pause():sleep(2);
            kill(0,SIGKILL);
            exit(0);
        }
    sleep(1);
    sprintf(str,"%d",getpid());
    execl("/usr/bin/pstree","pstree","-lp",str,NULL);
    kill(0,SIGKILL);
    exit(0);
}
$ gcc 2015-2_LSO-1_3D_AB_VK.c -o 2015-2_LSO-1_3D_AB_VK
$ ln 2015-2_LSO-1_3D_AB_VK 3D
$ ./3D 7
...
```

Pregunta 7 **(2 puntos – 10 min.)** ¿Cuál será el valor de la variable `value` si el proceso padre se ejecuta antes del proceso hijo? ¿Y si primero se ejecuta el proceso hijo y después el proceso padre?

```
$ cat fork_pointer.c

#include <stdio.h>
#include <unistd.h>

int value;

int
main(void)
{
    int *pval = &value;

    *pval=fork();
    printf("pid=%d, value=%d\n", getpid(), value);
}
```

Pregunta 8 **(2 puntos – 10 min.)** Explique que significa “la tubería rota” y cuando sucede la situación EOF con la tubería.

La práctica ha sido preparada por VK
con LibreOffice Writer en Linux Mint 17.2 Rafaela.

Profesor del curso: (0781) V. Khlebnikov

Pando, 11 de septiembre de 2015