PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ
FACULTAD DE CIENCIAS E INGENIERÍA

SISTEMAS OPERATIVOS
Examen 1
(Primer semestre de 2022)

Horarios 0781, 0782: prof. V. Khlebnikov

Duración: 3 horas
Nota: **La presentación, la ortografía y la gramática influirán en la calificación.**
Puntaje total: 20 puntos

Pregunta 1 (5 puntos – 30 min.)

El archivo de su respuesta debe estar en **INF239_078X_Ex1_P1_Buzón** del **Examen 1** en PAIDEIA **antes de las 08:45**. Por cada 3 minutos de retardo son -2 puntos.
El nombre de su archivo debe ser <*su_código_de_8_dígitos*>`_11.txt`. Por ejemplo, `20202912_11.txt`.

a) (3 puntos) Usando la siguiente distribución de Linux:

```
$ lsb_release -drc
Description:  Linux Mint 20.3
Release:      20.3
Codename:     una
```

se ejecuta la siguiente orden en *shell* y se obtiene la siguiente salida:

```
$ yes hola | head -3
hola
hola
hola
$
```

Indique, **en orden y en términos de las llamadas al sistema**, la secuencia de acciones que realizan los procesos para lograr la ejecución de la orden recibida.

Como anexo se les proporciona el código del programa `yes` de MINIX 2.0.4:

```
/*      yes 1.4 - print 'y' or argv[1] continuously.      Author: Kees J. Bot
 *                                                        15 Apr 1989
 */
#include <sys/types.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>

int main(int argc, char **argv)
{
        char *yes;
        static char y[] = "y";
        int n;

        yes= argc == 1 ? y : argv[1];

        n= strlen(yes);

        yes[n++]= '\n';

        while (write(1, yes, n) != -1) {}
        exit(1);
}
```

b) (2 puntos) Usted conoce el programa (implementado en MINIX 2.0.4, en este caso) cuyo código se presenta. ¿Cuál es su entrada? ¿Cuál es su salida? ¿Cuál es su nombre? Presente un ejemplo de su uso.

```
/*
 * ...
 *
 * Copyright (C) 1989 by Kenneth Almquist.  All rights reserved.
 * This file is part of ash, which is distributed under the terms specified
 * by the Ash General Public License.  See the file named LICENSE.
 */

#define main ...

#include "bltin.h"

#undef eflag


main(argc, argv)  char **argv; {
      register char **ap;
      register char *p;
      register char c;
      int count;
      int nflag = 0;
#ifndef eflag
      int eflag = 0;
#endif

      ap = argv;
      if (argc)
            ap++;
      if ((p = *ap) != NULL) {
            if (equal(p, "--")) {
                  ap++;
            }
            if (equal(p, "-n")) {
                  nflag++;
                  ap++;
            } else if (equal(p, "-e")) {
#ifndef eflag
                  eflag++;
#endif
                  ap++;
            }
      }
      while ((p = *ap++) != NULL) {
            while ((c = *p++) != '\0') {
                  if (c == '\\' && eflag) {
                        switch (*p++) {
                        case 'b':  c = '\b';  break;
                        case 'c':  return 0;         /* exit */
                        case 'f':  c = '\f';  break;
                        case 'n':  c = '\n';  break;
                        case 'r':  c = '\r';  break;
                        case 't':  c = '\t';  break;
                        case 'v':  c = '\v';  break;
                        case '\\':  break;           /* c = '\\' */
                        case '0':
                              c = 0;
                              count = 3;
                              while (--count >= 0 && (unsigned)(*p - '0') < 8)
                                    c = (c << 3) + (*p++ - '0');
                              break;
                        default:
                              p--;
                              break;
                        }
                  }
                  putchar(c);
            }
```

Es lo mismo que:

```
int main(argc, argv)
int argc;
char **argv;
{ ... }
```

O simplemente,

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

```
            if (*ap)
                putchar(' ');
        }
        if (! nflag)
            putchar('\n');
        return 0;
}
```

Preparado por VK
con LibreOffice Writer en Linux Mint 20.3 “Una”

Profesor del curso: V. Khlebnikov

Lima, 18 de mayo de 2022

PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ
FACULTAD DE CIENCIAS E INGENIERÍA

SISTEMAS OPERATIVOS
Examen 1
(Primer semestre de 2022)

Horarios 0781, 0782: prof. V. Khlebnikov

Duración: 3 horas
Nota: **La presentación, la ortografía y la gramática influirán en la calificación.**
Puntaje total: 20 puntos

Pregunta 2 (5 puntos – 30 min.)

El archivo de su respuesta debe estar en **INF239_078X_Ex1_P2_Buzón** del **Examen 1** en PAIDEIA **antes de las 09:30**. Por cada 3 minutos de retardo son -2 puntos.
El nombre de su archivo debe ser <*su_código_de_8_dígitos*>`_12.txt`. Por ejemplo, `20202912_12.txt`.

a) (2 puntos) Si preparamos un programa de una sola línea, un programa correcto que no hace nada, lo compilamos y ejecutamos:

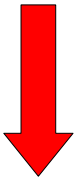
```
$ cat -n foo.c
     1 int main(int argc, char **argv) { }

$ date
lun 16 may 2022 22:06:27 -05

$ gcc foo.c

$ ls -l a.out
-rwxrwxr-x 1 vk vk 16464 may 16 22:06 a.out

$ ./a.out
$
```

No hay errores ni del compilador, ni del enlazador (*linker*), ni de la ejecución. Y el programa ejecutable es de 16464 bytes. Se sabe que en el directorio `/usr/bin/` se guardan los programas ejecutables. Entonces,

```
$ find /usr/bin -type f -size -16464c -ls | wc -l
1087
```

dice que hay 1087 programas ejecutables del menor tamaño. Algunos de unos decenas de bytes solamente. Explique cómo es posible y justifique con un ejemplo.

< ***Pase a la siguiente página*** >



b) (3 puntos) Considere el siguiente código de un programa de MINIX 2.0.4:

```
1  /* [redacted]    Authors: Andy Tanenbaum & Michiel Huisjes */
2
3  #define NEW 1
4
5  #include <sys/types.h>
6  #include <sys/times.h>
7  #include <limits.h>
8  #include <time.h>
9  #include <signal.h>
10 #include <stdlib.h>
11 #include <unistd.h>
12 #include <sys/wait.h>
13 #include <minix/minlib.h>
14 #include <stdio.h>
15
16 /* -DNEW prints time to 0.01 sec. */
17 #ifdef NEW
18 #define HUNDREDTHS 1
19 #endif
20
21 char **args;
22 char *name;
23
24 int digit_seen;
25 char a[] = "        . \0";
26
27 _PROTOTYPE(int main, (int argc, char **argv));
28 _PROTOTYPE(void print_time, (clock_t t));
29 _PROTOTYPE(void twin, (int n, char *p));
30 _PROTOTYPE(void execute, (void));
31
32 int main(argc, argv)
33 int argc;
34 char *argv[];
35 {
36
37   struct tms pre_buf, post_buf;
38   int status, pid;
39 #if _VMD_EXT
40   struct timeval [redacted]_time, [redacted]_time;
41 #else
42   time_t [redacted]_time, [redacted]_time;
43 #endif
44   clock_t real_time;
45
46   if (argc == 1) exit(0);
47
48   args = &argv[1];
49   name = argv[1];
50
51   /* [redacted] */
52 #if _VMD_EXT
53   (void) sysutime(UTIME_TIMEOFDAY, &[redacted]_time);
54 #else
55   (void) time(&[redacted]_time);
56 #endif
57
58   /* Fork off child. */
59   if ((pid = fork()) < 0) {
60     std_err("Cannot fork\n");
61     exit(1);
62   }
63   if (pid == 0) execute();
64
65   /* [redacted] */
66   signal(SIGINT, SIG_IGN);
67   signal(SIGQUIT, SIG_IGN);
68
69   do {
70     times(&pre_buf);
71   } while (wait(&status) != pid);
72 #if _VMD_EXT
73   (void) sysutime(UTIME_TIMEOFDAY, &[redacted]_time);
74   real_time = ([redacted]_time.tv_sec - [redacted]_time.tv_sec) * CLOCKS_PER_SEC
75     + ([redacted]_time.tv_usec - [redacted]_time.tv_usec) * CLOCKS_PER_SEC / 1000000;
76 #else
77   (void) time(&[redacted]_time);
78   real_time = ([redacted]_time - [redacted]_time) * CLOCKS_PER_SEC;
79 #endif
80
```

```
 81   if ((status & 0377) != 0) std_err("Command terminated abnormally.\n");
 82   times(&post_buf);
 83
 84   /* Print results. -DNEW enables time on one line to 0.01 sec */
 85 #ifndef NEW
 86   std_err("[redacted]");
 87   print_time(real_time);
 88   std_err("\n[redacted]");
 89   print_time(post_buf.tms_cutime - pre_buf.tms_cutime);
 90   std_err("\n[redacted]");
 91   print_time(post_buf.tms_cstime - pre_buf.tms_cstime);
 92   std_err("\n");
 93 #else
 94   print_time(real_time);
 95   std_err("[redacted]");
 96   print_time(post_buf.tms_cutime - pre_buf.tms_cutime);
 97   std_err("[redacted]");
 98   print_time(post_buf.tms_cstime - pre_buf.tms_cstime);
 99   std_err("[redacted]");
100 #endif
101   return((status & 0377) ? -1 : (status >> 8));
102 }
103
104 void print_time(t)
105 register clock_t t;
106 {
107 /* Print the time 't' in hours: minutes: seconds.  't' is in ticks. */
108
109   int hours, minutes, seconds, hundredths, i;
110
111   digit_seen = 0;
112   for (i = 0; i < 8; i++) a[i] = ' ';
113   hours = (int) (t / ((clock_t) 3600 * CLOCKS_PER_SEC));
114   t -= (clock_t) hours * 3600 * CLOCKS_PER_SEC;
115   minutes = (int) (t / ((clock_t) 60 * CLOCKS_PER_SEC));
116   t -= (clock_t) minutes * 60 * CLOCKS_PER_SEC;
117   seconds = (int) (t / CLOCKS_PER_SEC);
118   t -= (clock_t) seconds * CLOCKS_PER_SEC;
119   hundredths = (int) (t * 100 / CLOCKS_PER_SEC);
120
121   if (hours) {
122     twin(hours, &a[0]);
123     a[2] = ':';
124   }
125   if (minutes || digit_seen) {
126     twin(minutes, &a[3]);
127     a[5] = ':';
128   }
129   if (seconds || digit_seen)
130     twin(seconds, &a[6]);
131   else
132     a[7] = '0';
133   a[9] = hundredths / 10 + '0';
134 #ifdef HUNDREDTHS       /* tenths used to be enough */
135   a[10] = hundredths % 10 + '0';
136 #endif
137   std_err(a);
138 }
139
140 void twin(n, p)
141 int n;
142 char *p;
143 {
144   char c1, c2;
145   c1 = (n / 10) + '0';
146   c2 = (n % 10) + '0';
147   if (digit_seen == 0 && c1 == '0') c1 = ' ';
148   *p++ = c1;
149   *p++ = c2;
150   if (n > 0) digit_seen = 1;
151 }
152
153 void execute()
154 {
155   execvp(name, args);
156   std_err("Cannot execute ");
157   std_err(name);
158   std_err("\n");
159   exit(-1);
160 }
```

Explique cómo funciona durante la ejecución del programa el bloque de las líneas 69 – 71.

Explique y justifique las acciones que se realizan en las líneas 81 y 101.

¿Cuál es el nombre de este programa? ¿Qué resultado produce este programa? Presente un ejemplo de su ejecución en Linux.

Preparado por VK
con LibreOffice Writer en Linux Mint 20.3 “Una”

Profesor del curso: V. Khlebnikov

Lima, 18 de mayo de 2022

PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ
FACULTAD DE CIENCIAS E INGENIERÍA

SISTEMAS OPERATIVOS
Examen 1
(Primer semestre de 2022)

Horarios 0781, 0782: prof. V. Khlebnikov

Duración: 3 horas
Nota: **La presentación, la ortografía y la gramática influirán en la calificación.**
Puntaje total: 20 puntos

Pregunta 3 (5 puntos – 30 min.)

El archivo de su respuesta debe estar en **INF239_078X_Ex1_P3_Buzón** del **Examen 1** en PAIDEIA **antes de las 10:15**. Por cada 3 minutos de retardo son -2 puntos.
El nombre de su archivo debe ser <*su_código_de_8_dígitos*>`_13.txt`. Por ejemplo, `20202912_13.txt`.

(AB) Considere el siguiente programa `counter.c`:

```
 4 int counters[2];
 5
 6 void *
 7 th_func(void *arg)
 8 {
 9     int *thread_num, th_num, i;
10
11     thread_num = (int *)arg; th_num = *thread_num;
12
13     for (i=0; i<10; i++) {
14         printf("thread %d: iteration #%d\n", th_num, i);
15         counters[th_num] = counters[1-th_num] + 1;
16     }
17 }
18
19 int
20 main(int argc, char **argv)
21 {
22     pthread_t th1, th2;
23     int th1_arg, th2_arg;
24
25     th1_arg = 0; th2_arg = 1;
26
27     pthread_create(&th1, NULL, th_func, (void *)&th1_arg);
28     pthread_create(&th2, NULL, th_func, (void *)&th2_arg);
29
30     pthread_join(th1, NULL);
31     pthread_join(th2, NULL);
32
33     printf("counters[0]=%d, counters[1]=%d\n", counters[0], counters[1]);
34 }
```

(0,5 puntos) ¿En qué caso se obtienen los valores impresos 1 y 2?

(0,5 puntos) ¿En qué caso se obtienen los valores impresos 2 y 1?

(1 punto) ¿En qué caso se obtienen los valores impresos 3 y 2?

Considere el siguiente código:

```
struct _sb {
    pthread_cond_t cond;
    pthread_mutex_t mutex;
    int runners;
};

typedef struct {
    int maxcnt;
    struct _sb sb[2];
    struct _sb *sbp;
} b_t;


int
foobar(b_t *bp, int count)
{
    int i;

    if (count <= 1) {
        printf("Error: numero de hilos debe ser mayor que 1\n");
        exit(-1);
    }

    bp->maxcnt = count;
    bp->sbp = &bp->sb[0];
    for(i=0; i<2; i++) {
        struct _sb *sbp = &(bp->sb[i]);
        sbp->runners = count;
        pthread_mutex_init(&sbp-> mutex, NULL);
        pthread_cond_init(&sbp->cond, NULL);
    }
    return(0);
}

int
foobaz(b_t *bp)
{
    struct _sb *sbp = bp->sbp;
    pthread_mutex_lock(&sbp->mutex);
    if (sbp->runners == 1) {
        if (bp->maxcnt != 1) {
            sbp->runners = bp->maxcnt;
            bp->sbp = (bp->sbp = &bp->sb[0])? &bp->sb[1] : &bp->sb[0];
            pthread_cond_broadcast(&sbp->cond);
        }
    } else {
        sbp->runners--;
        while (sbp->runners != bp->maxcnt)
        pthread_cond_wait(&sbp->cond, &sbp->mutex);
    }
    pthread_mutex_unlock(&sbp->mutex);
}
```

(2 puntos) ¿Cuál es el objetivo de cada una de estas funciones? ¿Cuáles serían los nombres más apropiados para cada una de ellas?

(1 punto) Indique ¿en qué líneas y cómo se debería invocar a estas funciones para obtener el resultado correcto (10 y 10) con las cantidades de iteraciones de cada hilo en el código del programa `counter.c` que está al inicio de la hoja de esta pregunta?

Preparado por VK
con LibreOffice Writer en Linux Mint 20.3 “Una”

Profesor del curso: V. Khlebnikov

Lima, 18 de mayo de 2022

PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ
FACULTAD DE CIENCIAS E INGENIERÍA

SISTEMAS OPERATIVOS
Examen 1
(Primer semestre de 2022)

Horarios 0781, 0782: prof. V. Khlebnikov

Duración: 3 horas
Nota: **La presentación, la ortografía y la gramática influirán en la calificación.**
Puntaje total: 20 puntos

Pregunta 4 (5 puntos – 30 min.)

El archivo de su respuesta debe estar en **INF239_078X_Ex1_P4_Buzón** del **Examen 1** en PAIDEIA **antes de las 11:00**. Por cada 3 minutos de retardo son -2 puntos.
El nombre de su archivo debe ser <*su_código_de_8_dígitos*>_14.txt. Por ejemplo, 20202912_14.txt.

(AB) El problema de la cena de los filósofos. La siguiente solución al problema de los filósofos, empleando monitores, es incorrecta. Muestre un caso por la que falla y proponga una solución.

```
MONITOR CINCO_FILOSOFOS
type
    tipo_estado = (meditando, esperando_comer, comiendo);
var
    estado: array [0..4] of tipo_estado;
    espera: array [0..4] of condition;
    cont: integer;

procedure INTENTAR_COMER(j: integer)
begin
    if (estado[(j+1) mod 5] = comiendo) or (estado[(j-1) mod 5] = comiendo)
    then wait(espera[j]);
    else estado[j] := comiendo;
end;

procedure DEJAR_COMER(j : integer)
begin
    estado[j] := meditando;
    if (estado[(j+1) mod 5] = esperando_comer) and (estado[(j+2) mod 5] <> comiendo)
    then begin
             estado[(j+1) mod 5)] := comiendo;
             send(espera[(j+1) mod 5]);
         end;
    if (estado[(j-1) mod 5] = esperando_comer) and (estado[(j-2) mod 5] <> comiendo)
    then begin
             estado[(j-1) mod 5)] := comiendo;
             send(espera[(j-1) mod 5]);
         end;
end;

begin { monitor, su inicialización }
    for cont := 0 to 4 do
        estado[cont] := esperando_comer;
end;

{ fin del monitor }

-------------------------------------------------------------------------------
```

```
procedure filosofo(f: integer)
begin
    repeat
        meditar;
        INTENTAR_COMER(f);
        comer;
        DEJAR_COMER(f);
    until false
end;

begin { cuerpo del programa principal }
{ La palabra cobegin indica el comienzo de la ejecución concurrente de los procesos
que se señalan hasta la sentencia coend }
    cobegin
        filosofo(0);
        filosofo(1);
        filosofo(2);
        filosofo(3);
        filosofo(4);
    coend;
end.
```

Preparado por VK
con LibreOffice Writer en Linux Mint 20.3 “Una”

Profesor del curso: V. Khlebnikov

Lima, 18 de mayo de 2022