PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ
FACULTAD DE CIENCIAS E INGENIERÍA

SISTEMAS OPERATIVOS

1ra práctica (tipo a)
(Segundo semestre de 2017)

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 (8 puntos – 40 min.) Se prepara la siguiente orden:

```
$ cat | rev | tee r | head -n 1 | wc -c
```

donde los códigos de programas involucrados son los siguientes:

```
$ cat -n cat.c
  1  /* cat - concatenates files            Author: Andy Tanenbaum */
  2
  3  extern int errno; /*DEBUG*/
  4
  5  #include "blocksize.h"
  6  #include "stat.h"
  7
  8  #define BUF_SIZE      512
  9  int unbuffered;
 10  char buffer[BUF_SIZE];
 11  char *next = buffer;
 12
 13  main(argc, argv)
 14  int argc;
 15  char *argv[];
 16  {
 17    int i, k, m, fd1;
 18    char *p;
 19    struct stat sbuf;
 20
 21    k = 1;
 22    /* Check for the -u flag -- unbuffered operation. */
 23    p = argv[1];
 24    if (argc >=2 && *p == '-' && *(p+1) == 'u') {
 25          unbuffered = 1;
 26          k = 2;
 27    }
 28
 29    if (k >= argc) {
 30          copyfile(0, 1);
 31          flush();
 32          exit(0);
 33    }
 34
 35    for (i = k; i < argc; i++) {
 36          if (argv[i][0] == '-' && argv[i][1] == 0) {
 37                  fd1 = 0;
 38          } else {
 39                  fd1 = open(argv[i], 0);
 40                  if (fd1 < 0) {
 41                          std_err("cat: cannot open "); std_err(argv[i]); std_err("\n");
 42                          continue;
 43                  }
 44          }
 45          copyfile(fd1, 1);
 46          if (fd1 != 0) close(fd1);
 47    }
 48    flush();
 49    exit(0);
 50  }
 51
```

```
 52  copyfile(fd1, fd2)
 53  int fd1, fd2;
 54  {
 55    int n, j, m;
 56    char buf[BLOCK_SIZE];
 57
 58    while (1) {
 59          n = read(fd1, buf, BLOCK_SIZE);
 60          if (n < 0) quit();
 61          if (n == 0) return;
 62          if (unbuffered) {
 63                  m = write(fd2, buf, n);
 64                  if (m != n) quit();
 65          } else {
 66                  for (j = 0; j < n; j++) {
 67                          *next++ = buf[j];
 68                          if (next == &buffer[BUF_SIZE]) {
 69                                  m = write(fd2, buffer, BUF_SIZE);
 70                                  if (m != BUF_SIZE) quit();
 71                                  next = buffer;
 72                          }
 73                  }
 74          }
 75    }
 76  }
 77
 78  flush()
 79  {
 80    if (next != buffer)
 81          if (write(1, buffer, next - buffer) <= 0) quit();
 82  }
 83
 84  quit()
 85  {
 86    perror("cat");
 87    exit(1);
 88  }
```

```
$ cat -n rev.c
  1  /* rev - reverse an ASCII line    Authors: Paul Polderman & Michiel Huisjes */
  2
  3  #include "blocksize.h"
  4
  5  #ifndef NULL
  6  #define NULL    0
  7  #endif
  8
  9  #ifndef EOF
 10  #define EOF     ((char) -1)
 11  #endif
 12
 13  int fd;                 /* File descriptor from file currently being read */
 14
 15  main(argc, argv)
 16  int argc;
 17  char *argv[];
 18  {
 19    register unsigned short i;
 20
 21    if (argc == 1) {      /* If no arguments given, take stdin as input */
 22          fd = 0;
 23          rev();
 24          exit(0);
 25    }
 26    for (i = 1; i < argc; i++) {  /* Reverse each line in arguments */
 27          if ((fd = open(argv[i], 0)) < 0) {
 28                  std_err("Cannot open "); std_err(argv[i]); std_err("\n");
 29                  continue;
 30          }
 31          rev();
 32          close(fd);
 33    }
 34    exit(0);
 35  }
 36
 37  rev()
 38  {
 39    char output[BLOCK_SIZE];      /* Contains a reversed line */
 40    register unsigned short i;    /* Index in output array */
 41
 42    do {
 43          i = BLOCK_SIZE - 1;
```

```
44          while ((output[i] = nextchar()) != '\n' && output[i] != EOF)
45                  i--;
46          write(1, &output[i + 1], BLOCK_SIZE - 1 - i);   /* Write reversed line*/
47          if (output[i] == '\n')                          /* and write a '\n' */
48                  write(1, "\n", 1);
49    } while (output[i] != EOF);
50  }
51
52  char buf[BLOCK_SIZE];
53  nextchar()              /* Does a sort of buffered I/O */
54  {
55    static int n = 0;     /* Read count */
56    static int i;         /* Index in input buffer to next character */
57
58    if (--n <= 0) {       /* We've had this block. Read in next block */
59          n = read(fd, buf, BLOCK_SIZE);
60          i = 0;          /* Reset index in array */
61    }
62    return((n <= 0) ? EOF : buf[i++]);    /* Return -1 on EOF */
63  }
```

```
$ cat -n tee.c
 1  /* tee - pipe fitting           Author: Paul Polderman */
 2
 3  #include "blocksize.h"
 4  #include "signal.h"
 5
 6  #define MAXFD   18
 7
 8  int fd[MAXFD];
 9
10  main(argc, argv)
11  int argc;
12  char **argv;
13  {
14    char iflag = 0, aflag = 0;
15    char buf[BLOCK_SIZE];
16    int i, s, n;
17
18    argv++; --argc;
19    while (argc > 0 && argv[0][0] == '-') {
20          switch (argv[0][1]) {
21                  case 'i':       /* Interrupt turned off. */
22                          iflag++;
23                          break;
24                  case 'a':       /* Append to outputfile(s),
25                                   * instead of overwriting them.
26                                   */
27                          aflag++;
28                          break;
29                  default:
30                          std_err("Usage: tee [-i] [-a] [files].\n");
31                          exit(1);
32          }
33          argv++;
34          --argc;
35    }
36    fd[0] = 1;    /* Always output to stdout. */
37    for (s = 1; s < MAXFD && argc > 0; --argc, argv++) {
38          if ((fd[s] = open(*argv, 2)) < 0 &&
39                                  (fd[s] = creat(*argv, 0666)) < 0) {
40                  std_err("Cannot open output file: "); std_err(*argv); std_err("\n");
41                  exit(2);
42          }
43          s++;
44    }
45
46    if (iflag) signal(SIGINT, SIG_IGN);
47    for (i = 1; i < s; i++) {     /* Don't lseek stdout. */
48          if (aflag) lseek(fd[i], 0L, 2);
49    }
50
51    while ((n = read(0, buf, BLOCK_SIZE)) > 0) {
52          for (i = 0; i < s; i++)
53                  write(fd[i], buf, n);
54    }
55
56    for (i = 0; i < s; i++)               /* Close all fd's */
57          close(fd[i]);
58    exit(0);
59  }
```

```
$ cat -n head.c
```

```
  1  /* head - print the first few lines of a file   Author: Andy Tanenbaum */
  2
  3  #include "stdio.h"
  4
  5  #define DEFAULT 10
  6
  7  char buff[BUFSIZ];
  8  char lbuf[256];
  9
 10  main(argc, argv)
 11  int argc;
 12  char *argv[];
 13  {
 14    int n, k, nfiles;
 15    char *ptr;
 16
 17    /* Check for flag.  Only flag is -n, to say how many lines to print. */
 18    setbuf(stdout, buff);
 19    k = 1;
 20    ptr = argv[1];
 21    n = DEFAULT;
 22    if (*ptr++ == '-') {
 23          k++;
 24          n = atoi(ptr);
 25          if (n <= 0) usage();
 26    }
 27    nfiles = argc - k;
 28
 29    if (nfiles == 0) {
 30          /* Print standard input only. */
 31          do_file(n);
 32          fflush(stdout);
 33          exit(0);
 34    }
 35
 36    /* One or more files have been listed explicitly. */
 37    while (k < argc) {
 38          fclose(stdin);
 39          if (nfiles > 1) prints("==> %s <==\n", argv[k]);
 40          if (fopen(argv[k], "r") == NULL)
 41                  prints("head: cannot open %s\n", argv[k]);
 42          else
 43                  do_file(n);
 44          k++;
 45          if (k < argc) prints("\n");
 46    }
 47    fflush(stdout);
 48    exit(0);
 49  }
 50
 51
 52  do_file(n)
 53  int n;
 54  {
 55    /* Print the first 'n' lines of a file. */
 56    while (n--) do_line();
 57  }
 58
 59
 60  do_line()
 61  {
 62  /* Print one line. */
 63
 64    char c, *cp;
 65    cp = lbuf;
 66    while ( (c = getc(stdin)) != '\n') *cp++ = c;
 67    *cp++ = '\n';
 68    *cp++ = 0;
 69    prints("%s",lbuf);
 70  }
 71
 72
 73  usage()
 74  {
 75    std_err("Usage: head [-n] [file ...]\n");
 76    exit(1);
 77  }
```

```
$ cat -n wc.c
```

```
  1  /* wc - count lines, words and characters       Author: David Messer */
```

```
 2
 3  #include "stdio.h"
 4  #define isdigit(c) (c >= '0' && c <= '9')
 5  #define isspace(c) (c == ' ' || c == '\t' || c == '\n' || c == '\f' || c == '\r')
 6
 7  /*
 8   *
 9   *      Usage:  wc [-lwc] [names]
10   *
11   *              Flags:
12   *                      l - count lines.
13   *                      w - count words.
14   *                      c - count characters.
15   *
16   *              Flags l, w, and c are default.
17   *              Words are delimited by any non-alphabetic character.
18   *
19   *  Released into the PUBLIC-DOMAIN 02/10/86
20   *
21   *      If you find this program to be of use to you, a donation of
22   *      whatever you think it is worth will be cheerfully accepted.
23   *
24   *      Written by: David L. Messer
25   *                              P.O. Box 19130, Mpls, MN,  55119
26   *      Program (heavily) modified by Andy Tanenbaum
27   */
28
29
30  int lflag;                      /* Count lines */
31  int wflag;                      /* Count words */
32  int cflag;                      /* Count characters */
33
34  long lcount;                    /* Count of lines */
35  long wcount;                    /* Count of words */
36  long ccount;                    /* Count of characters */
37
38  long ltotal;                    /* Total count of lines */
39  long wtotal;                    /* Total count of words */
40  long ctotal;                    /* Total count of characters */
41
42  main(argc, argv)
43  int argc;
44  char *argv[];
45  {
46    int k;
47    char *cp;
48    int tflag, files;
49    int i;
50
51    /* Get flags. */
52    files = argc - 1;
53    k = 1;
54    cp = argv[1];
55    if (*cp++ == '-') {
56          files--;
57          k++;                      /* points to first file */
58          while (*cp != 0) {
59                  switch (*cp) {
60                          case 'l':       lflag++;        break;
61                          case 'w':       wflag++;        break;
62                          case 'c':       cflag++;        break;
63                          default:        usage();
64                  }
65                  cp++;
66          }
67    }
68
69    /* If no flags are set, treat as wc -lwc. */
70    if(!lflag && !wflag && !cflag) {
71          lflag = 1; wflag = 1; cflag = 1;
72    }
73
74    /* Process files. */
75    tflag = files >= 2;           /* set if # files > 1 */
76
77    /* Check to see if input comes from std input. */
78    if (k >= argc) {
79          count();
80          if(lflag) printf(" %6D", lcount);
81          if(wflag) printf(" %6D", wcount);
82          if(cflag) printf(" %6D", ccount);
83          printf(" \n");
84          fflush(stdout);
```

```
 85          exit(0);
 86    }
 87
 88    /* There is an explicit list of files.  Loop on files. */
 89    while (k < argc) {
 90          fclose(stdin);
 91          if (fopen(argv[k], "r") == NULL) {
 92                  std_err("wc: cannot open "); std_err(argv[k]); std_err("\n");
 93                  k++;
 94                  continue;
 95          } else {
 96                  /* Next file has been opened as std input. */
 97                  count();
 98                  if(lflag) printf(" %6D", lcount);
 99                  if(wflag) printf(" %6D", wcount);
100                  if(cflag) printf(" %6D", ccount);
101                  printf(" %s\n", argv[k]);
102          }
103          k++;
104    }
105
106    if(tflag) {
107          if(lflag) printf(" %6D", ltotal);
108          if(wflag) printf(" %6D", wtotal);
109          if(cflag) printf(" %6D", ctotal);
110          printf(" total\n");
111    }
112
113    fflush(stdout);
114    exit(0);
115  }
116
117  count()
118  {
119    register int c;
120    register int word = 0;
121
122    lcount = 0; wcount = 0; ccount = 0L;
123
124    while((c = getc(stdin)) > 0) {
125          ccount++;
126          if(isspace(c)) {
127                  if(word) wcount++;
128                  word = 0;
129          } else {
130                  word = 1;
131          }
132          if (c == '\n' || c == '\f') lcount++;
133    }
134    ltotal += lcount; wtotal += wcount; ctotal += ccount;
135  }
136
137  usage()
138  {
139    std_err("Usage: wc [-lwc] [name ...]\n");
140    exit(1);
141  }
```

(a) (2 puntos – 10 min.) Si después de ingresar la orden mencionada y completarla con la tecla Enter solamente, abrir otro terminal e imprimir con **`ps`** los estados de los procesos creados por la orden mencionada, ¿cuál será el árbol de los procesos? (Use los números para PID a partir de 50.)

(b) (3 puntos – 15 min.) ¿Cuáles son los valores (en números de líneas de código) de los *Program Counters (PC)* en cada proceso en el momento cuando se obtuvo el árbol de los procesos? ¿Cuáles son los estados de cada uno de los procesos? Explique la causa en términos de llamadas al sistema.

(c) (3 puntos – 15 min.) Si en el 1er terminal ingresar las siguientes 5 líneas:

```
gnik nus eht semoc ereh
gnik nus eht semoc ereh
gnihgual sydobyreve
yppah sydobyreve
gnik nus eht semoc ereh <Ctrl+D>
```

¿Cuál será el resultado de ejecución y el contenido del archivo creado?

Pregunta 2 **(8 puntos – 40 min.)**

(a) (2 puntos – 10 min.) ¿Cuál será la salida como resultado de ejecución del siguiente programa?

```
$ cat forkbasic.c
#include <sys/types.h>
#include <unistd.h>

int
main(void)
{
  char i;

  for (i='A'; i<='Z'; i++) {
    if (i == 'M') fork();
    write(1, &i, 1);
  }
  i = '\n';
  write(1, &i, 1);
}
```

(b) (3 puntos – 15 min.) ¿Si en el programa anterior incluir `sleep(1)` después del primer `write()`, ¿cuál será la salida como resultado de ejecución del programa?

(c) (3 puntos – 15 min.) Y, una vez más, ¿cuál será la salida como resultado de ejecución del siguiente programa?

```
$ cat forkbasic3.c
#include <sys/types.h>
#include <unistd.h>
#include <stdio.h>

int
main(void)
{
  char i;

  for (i='A'; i<='Z'; i++) {
    if (i == 'M') fork();
    printf("%c",i);
  }
  printf("\n");
}
```

Pregunta 3 **(4 puntos – 10 min.)**

(a) (2 puntos – 10 min.) Suppose that initially `x = 0` and we run a program with three threads that do the following:

Thread A	**Thread B**	**Thread C**
`x *= 2;`	`x *= x;`	`x += 3;`

What are the possible final values of `x`?

(b) (2 puntos – 10 min.) Suppose that initially `x = 0` and we run a program with three threads that do the following:

Thread A	**Thread B**	**Thread C**
`x += 2;`	`x += 3;`	`x += 4;`

What are the possible final values of `x`?

La práctica ha sido preparada por VK
con LibreOffice Writer en Linux Mint 18.2 Sonya.

Profesor del curso: (0781) V. Khlebnikov

Pando, 15 de septiembre de 2017