Primer Certamen
Tiempo 100 min. Responder un problema por página

1.- (30 puntos Shell Script) A través de un script shell, se desea obtener experimentalmente el tamaño de un sector físico en disco. Para ello se pide usar el comando df, el cual reporta el uso de disco del sistema de archivos. Se sugiere usar opción -BK - -output=used para obtener la cantidad de disco usado en unidades de 1024 bytes (1K). Un ejemplo de este comando es:

```
agustin@agustin:~$ df -BK --output=used hola.txt
      Used
133225036K
agustin@agustin:~$
```

Donde "hola.txt" es un archivo del sistema de archivos donde se desea conocer el tamaño del sector físico.
Luego se sugiere crear un archivo temporal de un tamaño pequeño y volver a ejecutar el comando previo. La diferencia del espacio de disco utilizado indicará el tamaño de sector físico de ese sistema de archivos.
Use pss.sh (por physical sector size) como nombre de su script. Éste usa como argumento el nombre de un archivo cuyo sistema de archivos se desea consultar (en el ejemplo es "hola.txt"). El script arroja como resultado el tamaño del sector físico de ese sistema de archivos.

Nota informativa que no se puede usar en su respuesta: Un comando del sistema para consultar por este valor es: $ lsblk -o PHY-SeC /dev/sda5
Donde /dev/sda5 es el dispositivo de bloques que se desea consultar.

```
#! /bin/bash
#pss <file>   return the size of a physical sector on the file devise.
file=$1;
IFS='K'   # inter field separator
set `df -BK --output=used $file | tail -n1`
init=$1
echo "Some text to the file" > $file$$
set `df -BK --output=used $file | tail -n1`
last=$1
echo `expr $last - $init` Kbytes
rm  $file$$
```

2.- (35 puntos) Se desea obtener en forma experimental la capacidad de una pipa, en otras palabras el tamaño del buffer usado por una pipa para alojar los datos que van de un proceso a otro. Haga un programa en C que muestre por pantalla la capacidad de la pipa creada por popen(...) y aquella creada por el llamado a pipe(...).

Nota informativa que no se puede usar en su respuesta: La capacidad de una pipe por defecto (default) es informada en la página man de pipe. Esta capacidad puede ser cambiada y obtenida usando el llamado a fcntl(…); sin embargo aquí se pide obtenerla experimentalmente.

Programa de apoyo (p2.child):

```
/*
 * wait without doing anything for 4 seconds.
 *  Created on: Sep 22, 2016
 *      Author: Agustín J. González
 * usage: p2.child
 */
```

```c
#include <unistd.h>
#include <stdlib.h>
#include <stdio.h>

int main (void) {
   sleep(4);
   getchar();
   exit(0);
}
```

Programa principal :

```c
#include <stdio.h>
#include <stdlib.h>
#include <signal.h>
#include <unistd.h>
int capacity;
static void sig_alarm_popen(int signo){
      printf ("Popen capacity= %d\n", capacity);
    exit (0);
}
static void sig_alarm_pipe(int signo){
      printf ("Pipe capacity= %d\n", capacity);
    exit (0);
}

int main(void)  {
    pid_t pid;
    FILE *pf;
    int pfd[2];
    int i, status;

    pid=fork();
    if (pid==0) { /*child*/
        FILE *pf;
        signal(SIGALRM, sig_alarm_popen);
        if ((pf = popen("./p2.child", "w")) == NULL) {
            perror("popen");
            exit(1);
        }
        capacity=0;
        alarm(1);
        while (fprintf(pf, "a")>0) {
            capacity++;
            alarm(1);
        }
    } else  {
        sleep(1);
        signal(SIGALRM, sig_alarm_pipe);
        if (pipe(pfd) < 0) {
            perror("pipe");
            exit(1);
        }
        if (fork()==0) {
            dup2(pfd[0], 0);
            close(pfd[1]);
            execl("./p2.child", "p2.child", (char *) 0);
            perror("exec");
            exit(127);
        }
        capacity=0;
```

```
        alarm(1);
        close(pfd[0]);
        while (write(pfd[1], "a", 1)>0) {
            capacity++;
            alarm(1);
        }
    }
}
```

3.- (35 puntos) hist(Y) es una función de Octave que crea un histograma de los valores del vector Y . En la tarea 2 su grupo implementó un algoritmo para detectar silencios en un flujo de audio. En esta pregunta usted debe crear el programa histoSV.cpp, el cual muestra dos histogramas: uno de la duración de los momentos de habla y otro, bajo el previo, de la duración de los silencios. Como algoritmo para detectar silencio en un paquete, compare la energía promedio de las muestras de ese paquete con un umbral dado. Si la energía supera el umbral, se trata de un paquete de voz. La sintaxis del programa es:
$ histoSV <archivo de audio> <tamaño de paquete> <umbral>
Al igual que en la tarea, el archivo contiene muestras de audio mono canal en formato PCM, little endian, con signo y de 16 bits. El tamaño de paquete es en muestras.

Nota: en Octave usted puede agregar un nuevo número ***num*** al vector X usando: X = [X, num];
Solución similar a ejemplo visto en clases:

```
/*
 * Voice and Silence durations histograms for Audio signals
 *
 *  Created on: Sep 22, 2016
 *      Author: Agustín J. González
 * It assumes mono audio file of 16-bit PCM samples, little-endian format,
sampled at 8 KHz
 * usage: histoSV <archivo de audio> <tamaño de paquete> <umbral>
 * Tested on Ubuntu 16.04 LTS with octave version 4.0.0 Copyright (C) 2015
 */

#include <unistd.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <sys/wait.h>
#include <fstream>  // ifstream , ofstream

using namespace std;

int main (int argc, char * argv[]) {
    double threshold;
    ifstream audioInFile;
    int packetSize;
    short *buff;
    pid_t pid;
    int pfd[2], status;
    FILE * octave;
    float vd=0, sd=0;  /* voice and silence periods in seconds */

    if (argc < 4) {
        printf("Usage: %s <input file> <packet size> <threshold> \n", argv[0]);
```

```
        exit(-1);
    }
    audioInFile.open (argv[1], ios::in | ios::binary);  // input audio file
    packetSize = atoi(argv[2]);
    threshold = atof(argv[3]);  //threshold of sensibility
    buff = new short[packetSize];
    pipe(pfd);
    pid=fork();
    if (pid == 0) {
        int junk;
        dup2(pfd[0], 0);
        close(pfd[1]);  /* close the write end of the pipe */
        junk = open("/dev/null", O_WRONLY);
        dup2(junk, 1);  /* throw away any message sent to screen*/
        dup2(junk, 2);  /* throw away any error message */
        execlp("octave", "octave", "-i", (char *) NULL);
        exit(-1);
    }
    close(pfd[0]);
    octave = fdopen(pfd[1], "w");  /* to use fprintf instead of just write */
    fprintf(octave, "V=[];\n S=[];\n");
    while (true) {
        double energy=0;
        int sampleRead;

        audioInFile.read((char *)buff, sizeof(short)*packetSize);
        if (audioInFile.eof()) break;
        sampleRead= audioInFile.gcount()/sizeof(short);
        for (int i=0; i <sampleRead; i++)
            energy += buff[i]*buff[i];
        energy /= sampleRead;
        if (energy > threshold){
            vd+=sampleRead*0.000125;
            if (sd > 0) { /* silence sequence finished */
                fprintf(octave, "S=[S, %f]; \n",sd);
                sd=0;
            }
        }
        else {
            sd+=sampleRead*0.000125;
            if (vd > 0) { /* speech sequence finished */
                fprintf(octave, "V=[V,%f]; \n",vd);
                vd=0;
            }
        }
      }
      audioInFile.close();
    if (sd > 0)  fprintf(octave, "S=[S, %f]; \n",sd);
    if (vd > 0)  fprintf(octave, "V=[V, %f]; \n",vd);
    fprintf(octave, "subplot (2,1,1);\n hist(V);\n  subplot(2,1,2);\n hist(S);\n
drawnow();\n");
    fflush(octave);
    getchar();
    fprintf(octave, "quit\n");  fflush(octave);
    waitpid(pid, &status, 0);
    exit(0);
}
```

Otras soluciones a problemas:

1)

Segunda solución:

```
! /bin/bash
#pss <file>   return the size of a physical sector on the file devise.
file=$1;
init=`df -BK --output=used $file | tail -n 1 | awk -FK '{print $1}'`
echo "Some text to the file" > $file$$
last=`df -BK --output=used $file | tail -n 1 | awk -FK '{print $1}'`
echo `expr $last - $init` Kbytes
rm  $file$$
```

Tercera solución:

```
#! /bin/bash
#pss <file>   return the size of a physical sector on the file devise.
file=$1
function usedSpace() {
  df -k --output=used $1 | tail -n 1
}
init=`usedSpace $file`
echo "Some text to the file" > $file$$
last=`usedSpace $file`
echo `expr $last - $init` Kbytes
rm  $file$$
```

3)

```
/*
 * Voice and Silence durations histograms for Audio signals
 *
 *  Created on: Sep 22, 2016
 *      Author: Agustín J. González
 * It assumes mono audio file of 16-bit PCM samples, little-endian format,
sampled at 8 KHz
 * usage: histoSV <archivo de audio> <tamaño de paquete> <umbral>
 * Tested on Ubuntu 16.04 LTS with octave version 4.0.0 Copyright (C) 2015
 */

#include <unistd.h>
#include <stdlib.h>
#include <fstream> // ifstream , ofstream

using namespace std;

int main (int argc, char * argv[]) {
  double threshold;
  ifstream audioInFile;
  int packetSize;
  short *buff;
  FILE * octave;
  float vd=0, sd=0; /* voice and silence periods in seconds */

  if (argc < 4) {
    printf("Usage: %s <input file> <packet size> <threshold> \n", argv[0]);
    exit(-1);
  }
  audioInFile.open (argv[1], ios::in | ios::binary); // input audio file
  packetSize = atoi(argv[2]);
```

```
  threshold = atof(argv[3]); //threshold of sensibility
  buff = new short[packetSize];
  octave = popen ("octave --no-gui", "w"); // octave version 4.0.0
  fprintf(octave, "V=[];\n S=[];\n");
      while (true) {
    double energy=0;
    int sampleRead;

    audioInFile.read((char *)buff, sizeof(short)*packetSize);
    if (audioInFile.eof()) break;
    sampleRead= audioInFile.gcount()/sizeof(short);
    for (int i=0; i <sampleRead; i++)
      energy += buff[i]*buff[i];
    energy /= sampleRead;
    if (energy > threshold){
      vd+=sampleRead*0.000125;
      if (sd > 0) { /* silence sequence finished */
        fprintf(octave, "S=[S, %f]; \n",sd);
        sd=0;
      }
    }
    else {
      sd+=sampleRead*0.000125;
      if (vd > 0) { /* speech sequence finished */
        fprintf(octave, "V=[V,%f]; \n",vd);
        vd=0;
      }
    }
  }
  audioInFile.close();
  if (sd > 0) fprintf(octave, "S=[S, %f]; \n",sd);
  if (vd > 0) fprintf(octave, "V=[V, %f]; \n",vd);
   fprintf(octave, "subplot (2,1,1);\n hist(V);\n   subplot(2,1,2);\n hist(S);\n
drawnow();\n");
  fflush(octave);
  getchar();
  fprintf(octave, "quit\n"); fflush(octave);
  pclose(octave);
}
```