**Primer Certamen**
**Tiempo 90 min. Responder un problema por página**

1.- (35 puntos, Shell Script) Para buscar cuentas de usuarios "hackeadas", un administrador quiere conocer los países y usuarios que han accedido a una máquina desde fuera de Chile el último tiempo. Para ello le pide crear el script shell "fa.sh" (por foreign access) que lista los países distintos de Chile y usuarios que han accedido a la máquina donde corre el script. La salida debería ser del tipo:

$ fa.sh
US alumno.estadounidense
FR alumna.francesa
..

Nota: aquí alumno.estadounidense y alumna.francesa son dos user names.

Ayuda:

* Considere la salida de comando ***last*** con opción "-a" (para listar dirección IP al final) y "-w" (para listar full user names). Así una línea de ***last*** sería:

cristian.lezana.13 pts/7 Mon Oct 23 17:22 - 17:22 (00:00) 201.241.41.46

* Considere el uso de wget con opción "-O -" (para enviar a salida estándar) y "-q" (para salida silenciosa "quiet") para acceder vía web a la base de datos ipinfo.io. Ésta entrega información asociada a una IP. Analice su salida al consultar por IP 193.252.122.73:

$ wget -q -O - ipinfo.io/193.252.122.73

{
"ip": "193.252.122.73",
"hostname": "www.orange.fr.inter.b0.fti.net",
"city": "",
"region": "",
"country": "FR",
"loc": "48.8582,2.3387",
"org": "AS24600 Orange S.A."
}

Para IPs en Chile, asociado a "country" aparecerá "CL".

```
#! /bin/bash
last -wa | while read username pts day month num time1 char time2 time3 IP
do
  IFS_OLD=$IFS                          #IFS es Internal Field Separator
  IFS=\"
  set -- `wget -q -O - ipinfo.io/$IP | grep country `
  IFS=$IFS_OLD                       # restablecer para el read del while
  if [ "$2" == "country" ]
  then
    if ! [ "$4" == "CL" ]
    then echo $4  $username
    fi
  fi
done
```

Otra solución posible:

```
#! /bin/bash
last -wa | while read username pts day month num time1 char time2 time3 IP
do
  set -- `wget -q -O - ipinfo.io/$IP | grep country | tr -d \",`
  if [ "$1" == "country:" ]
  then
    if ! [ "$2" == "CL" ]
    then echo $2  $username
    fi
  fi
done
Puntaje: 12 pts. por seleccionar IP y user name
         12 pts. Para cada IP obtener el país de procedencia
          6 pts. Seleccionar paises distintos a CL.
          5 pts. Mostrar país y username.
```

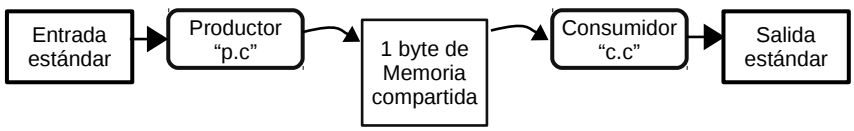
2.- (30 puntos) Desarrolle en C saturacion.c. Éste captura segmentos de audio en tiempo real desde el micrófono y, para cada uno, muestra en pantalla el número de muestras sobre un umbral. El audio debe ser mono-canal en formato PCM, little endian, con signo y de 16 bits (format=S16_LE). Los segmentos son de 512 muestras. Se pide utilizar arecord para grabar audio. Una invocación sería:
$saturacion  32000 /* aquí 32000 es el umbral,  arecord usa mismos argumentos que aplay */

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

int main(int agc, char * argv[]) {
    short s;   /* audio sample */
    int threshold = atoi(argv[1]);
    int i, saturated;
    FILE *pf;

    if ((pf = popen("arecord --format=S16_LE --quiet --file-type raw", "r")) == NULL)
        exit(-1);   /* La respuesta esperada pide solo especificar formato */
/* Hasta aquí 10 pts */
    while (1) {        /* cálculo de número de muestras para cada segmento 14 pts.*/
      saturated=0;
      for (i=0; i< 512; i++){
        if (fread(&s, sizeof(s), 1, pf) <= 0)
          exit(-1);
        if (s > threshold)
          saturated++;
      }
      printf("Number of saturated samples:%d\n", saturated);  /* 6 pts */
    }
}
```

3.- (35 puntos) Cree los programas p.c y c.c  (productor y consumidor).  p.c lee la entrada estándar de a un caracter a la vez, el cual almacena en un byte de memoria compartida.  El programa consumidor toma el caracter de la memoria compartida y lo muestra en su salida estándar. Ambos programas pueden funcionar en forma concurrente transfiriendo un número indefinido de caracteres.  Se pide usar biblioteca POSIX.



```c
/*p.c*/
#include <unistd.h>
#include <fcntl.h>
#include <sys/stat.h>
#include <sys/mman.h>
#include <semaphore.h>

int main(void) {
  const char *name = "/shm-C1_2s17";  /* file name*/
  const int SIZE=1;       /* file size */
  int shm_fd;/* file descriptor, from shm_open() */
  char *buff;   /* base address, from mmap()*/
  sem_t *empty, *full;

/* Creación de zona de memoria compartida: 5 pts. */
  shm_fd = shm_open(name, O_CREAT | O_RDWR, 0666);
  ftruncate(shm_fd, SIZE);
  buff = mmap(0, SIZE, PROT_READ | PROT_WRITE, MAP_SHARED, shm_fd, 0);
   /* Creación de dos semáforos: 5 pts. */
    /* first remove the semaphore if it already exists*/
  sem_unlink("EmptySem");
  /* create and initialize the semaphore */
 empty = sem_open("EmptySem", O_CREAT, 0666, 1);
  sem_unlink("FullSem");
  /* create and initialize the semaphore*/
  full = sem_open("FullSem", O_CREAT, 0666, 0);

  while (1) {   /*lee desde stdin un carácter a la vez con control de acceso: 5 pts. */
    sem_wait(empty);
    if (read(0,buff, 1)<=0)    break;
    sem_post(full);
  }
  *buff=0;;
  sem_post(full);
  sem_close(empty);   /* cierra sin remover : 1 pts. */
  sem_close(full);
  /* remove the mapped memory segment from the address space of the process */
  munmap(buff, SIZE);

  /* close the shared memory segment as if it was a file */
  close(shm_fd);
  return 0;
}
```

```c
/* c.c*/
#include <unistd.h>
#include <fcntl.h>
#include <sys/stat.h>
#include <sys/mman.h>
#include <semaphore.h>

int main(void) {
  const char *name = "/shm-C1_2s17";    /* file name*/
  const int SIZE = 1;             /* file size */
  int shm_fd;       /* file descriptor, from shm_open() */
  char *buff;/* base address, from mmap() */
  sem_t *empty, *full;
  /* Acceso a zona de memoria compartida: 5 pts. */
  /* open the shared memory segment as if it was a file */
  shm_fd = shm_open(name, O_RDONLY, 0666);

  /* map the shared memory segment to the address space of the process */
  buff = mmap(0, SIZE, PROT_READ, MAP_SHARED, shm_fd, 0);
/* Apertura de dos semáforos: 5 pts. */
  /* because this semaphore is created by producer, here it is just open */
  empty = sem_open("EmptySem", O_CREAT, 0666, 1); /* mode (666) and value are ignored*/
  /* because this semaphre is created by producer, here it is just open */
  full = sem_open("FullSem", O_CREAT, 0666, 0);

  while (1) {/*escribe en stdout un carácter a la vez con control de acceso: 5 pts. */
    sem_wait(full);
    write(1, buff, 1);
    if (*buff==0) break;
    sem_post(empty);
  }

  sem_close(empty);   /* cierra y remueve memoria y semáforos : 4 pts. */
  sem_unlink("EmptySem");
  sem_close(full);
  sem_unlink("FullSem");

  /* remove the mapped shared memory segment from the address space of the process */
  munmap(buff, SIZE);

  /* close the shared memory segment as if it was a file */
  close(shm_fd);

  /* remove the shared memory segment from the file system */
  shm_unlink(name);
  return 0;
}
```