Primer Certamen
Tiempo 100 min. Responder un problema por página

1.-
a) (20 puntos Shell Script) Cree el script shell is.sh (por image size) el cual recibe un elemento del directorio como parámetro y muestra por pantalla el tamaño en bytes y nombre del archivo si se trata de una imagen, en otros casos la salida indica "No es una imagen".
Ejemplo:
$ is.sh logo.jpg
16980 logo.jpg
$ is.sh logo.txt
No es imagen.

Ayuda: revise el comando "file" con opción -b. Revise el comando du -b filename

```
#! /bin/bash
# it shows a file size and name when it is an image.
if ( file -b $1 | grep image > /dev/null )
then
   du -b $1
else
   echo "No es imagen."
fi
```

b) (20 puntos Shell Script) Cree el script il.sh (por image list) el cual recibe como parámetro un directorio y lista el tamaño y nombre de todas las imágenes bajo ese directorio.

Ayuda: Suponga que se cuenta con el script is.sh descrito en a) y por lo tanto lo puede suar en su solución de b).

```
#! /bin/bash
#scan directory recursively and print size and name of every image
for i in $1/*
do
        if [ -d "$i" ]
        then
                $0 $i
        else
                result=`./is.sh $i`
           if !( echo $result  | grep "No es imagen" > /dev/null )
           then
               echo $result
           fi
        fi
done
```

2.- (30 puntos) Cree un programa en lenguaje C que permita determinar experimentalmente el tamaño de la tabla de descriptores.
Como estrategia se sugiere duplicar algún descriptor hasta que el llamado retorne error. Vea man dup. Para acceder a la variable global "errno" basta incluir el encabezado ~~error.h~~ errno.h (ver man errno).

```
#include <stdio.h>  /* for printf */
#include <stdlib.h> /* for exit */
#include <unistd.h> /* for dup */
#include <errno.h>  /* for errno */

int main(void)  {
   pid_t pid;
   int max=0;
   while (dup(0)>0)
      max ++;
   if (errno == EMFILE)
      printf("El tamaño de la tabla es %d \n", max+3);
      /* 3 is added to count for stdin stdout stderr */
   else
      printf("El error retornado es %d\n", errno);
   exit(0);
}
```

3.- (30 puntos) Cree el programa “grafica.c” en lenguaje C, éste pide ingresar una función por teclado y luego la grafica usando gnuplot. Ejemplo de ejecución:
$ grafica
Ingrese su función: sin(x)
(sin (x) fue ingresado por el usuario, lo previo fue escrito por el programa.)

y luego se muestra la gráfica de sin(x). Las funciones ingresadas deben ser compatibles con el comando plot de gnuplot.

```
#include <unistd.h>  /* dup2 */
#include <stdio.h>  /* printf */
#include <stdlib.h>  /* exit */
#include <sys/wait.h>  /* waitpid */

int main(void)  {
   pid_t pid;
   int pfd[2];
   int i, status;
   FILE * sd;
   char line[100];

   if (pipe(pfd) < 0) {
      perror("pipe");
      exit(1);
   }
   if ((pid = fork()) < 0) {
      perror("fork");
      exit(1);
   }
   if (pid == 0) {
      dup2(pfd[0], 0);
      close(pfd[1]);  /* close the write end off the pipe */
      execlp("gnuplot", "gnuplot", NULL);
      _exit(127);
   }

   close(pfd[0]);
   sd = fdopen(pfd[1], "w");
   printf("Ingrese su función: "); fflush(stdout);
   read (STDIN_FILENO, line, 100);
   fprintf(sd, "plot %s \n", line);
   fflush(sd);
   sleep(5);
   fprintf(sd, "exit\n"); fflush(sd);
   fclose(sd);
   waitpid(pid, &status, 0);
   exit(0);
}
```