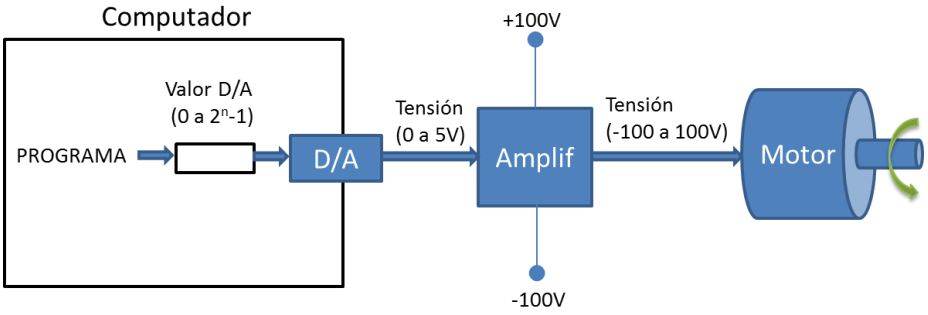
Control de Procesos en Tiempo Real

Examen Extraordinario - Junio 2012

4 puntos 1) Para el control de un motor de corriente alterna, se precisa escribir un programa que genere una señal senoidal con la frecuencia y amplitud deseadas. Para ello, se dispone del montaje siguiente:



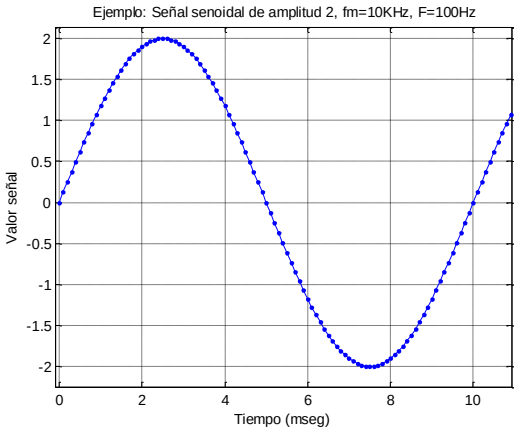
Las condiciones de funcionamiento son las siguientes:

- Se dispone de una librería con las siguientes funciones para la temporización y E/S:

```
// Inicialización de un temporizador que despierta a un hilo cada
// cierto tiempo (medido en microsegundos), mediante la activación de la
// condición c (asociada al mutex m)
void InitTemporizador(int tiempo_us,pthread_mutex_t* m,pthread_cond_t* c);

// Generación de una tensión analógica en el canal D/A deseado utilizando
// nBits : tensión de salida 0 a 5V para un valor de entrada 0 a 2^n-1.
void ConversionDA(int nCanal,int valor,int n);
```

- Se desea utilizar conversión D/A de 10 bits, y frecuencia de muestreo (f_m) de 10 KHz.
- Algoritmo para generación de una onda senoidal de una frecuencia (F) y amplitud (A) deseadas:
  - $\alpha=2.\pi.F/f_m$
  - $y_0=0$, $y_1=A*\alpha$
  - $y_k=-(a_1.y_{k-1}+a_2.y_{k-2})$, con: $a_1=-2.\cos(\alpha)$, $a_2=1$



- La generación de la onda senoidal se realiza en un hilo secundario, que ejecuta un bucle infinito en el cual, en cada pasada, se queda a la espera de la condición de la temporización y, una vez recibida, realiza el cálculo y la generación de la tensión de salida correspondiente al nuevo instante.
- El hilo principal se queda en un bucle solicitando por teclado nuevos valores de frecuencia (F) y amplitud (A). Sólo cuando se cambien ambos valores, el hilo principal los actualizará y colocará los valores iniciales correspondientes y_0 e y_1.
- Se deben proteger adecuadamente las variables compartidas.

Control de Procesos en Tiempo Real

Examen Extraordinario - Junio 2012

2) *3 puntos*

Realizar una función que reciba como parámetros:

- El nombre de un archivo.
- El directorio en que se encuentra el archivo.
- El número de datos (valores reales) a leer de dicho archivo.

A partir de dichos parámetros, la función deberá:

- Determinar, a partir de la extensión que tenga el nombre de archivo, si es de tipo texto o binario (tipo texto ≡ extensión ".txt", tipo binario ≡ extensión ".bin").
- Leer del archivo los datos deseados, según el formato detectado.
- Calcular y devolver la media y la desviación típica de los datos.

Escribir un ejemplo de llamada a esta función desde main() para calcular media y desviación de una tabla de 10 datos que se encuentre en el archivo "C:\Cptr\ejemplo.bin".

NOTA: No se puede suponer a priori un tamaño máximo de los datos, ni de los nombres de archivo y directorio.

3) *2 puntos*

Un servidor de páginas web (tipo http) implementa un socket servidor TCP que espera la conexión de clientes (navegadores web) en el puerto 80. Una vez realizada la conexión, el cliente puede enviar la cadena "GET /pagina_a_visitar.htm", y el servidor contestará con un mensaje que contiene el texto de dicha página.

Si, dentro de los primeros 100 caracteres del texto recibido por el cliente, se encuentra la cadena "404 Not Found", significa que la página solicitada no existe o no es accesible.

Realizar un programa que solicite por teclado la dirección IP de un servidor http, implemente el lado del cliente e indique por pantalla:

- Si se ha podido conectar con el servidor http en dicha dirección.
- En caso afirmativo, si existe en el mismo la página "/index.htm".

NOTA: Se puede utilizar para el socket cliente cualquier puerto IP (valor 0 en sin_port).

4) *1 punto*

Para el programa del ejercicio 1, explique **brevemente** cómo se realizaría la temporización utilizando temporizadores y señales POSIX , en lugar de con la función InitTemporizador().

Control de Procesos en Tiempo Real

Examen Extraordinario - Junio 2012

Paso de parámetros a main() desde línea de comandos:

```
int main(int argc,char* argv[])  // argv[1] es una cadena de caracteres para el
                                 // 1er parámetro, argv[2] para el 2°, etc.
```

Algunas funciones de C:

```
int atoi(const char* cadena);
double atof(const char* cadena);
double strtod(const char* cadena,char** final);

int strlen(const char* cadena);
char* strcpy(char* dst,const char* src);
char* strncpy(char* dst,const char* src,int n);
char* strcat(char* dst,const char* src);
char* strncat(char* dst,const char* src,int n);
int strcmp(const char* cad1,const char* cad2);
int strncmp(const char* cad1,const char* cad2,int n);
char* strchr(const char* cad,char c);
char* strstr(const char* cad,const char* busca);
int sprintf(char* dest,const char* fmt,...);
int sscanf(const char* src,const char* fmt,...);

FILE* fopen(const char* filename,const char* mode);
char* fgets(char* dest,int tam_max,FILE* f);
int fprintf(FILE* fid,const char* fmt,...);
int fscanf(FILE* fid,const char* fmt,...);
int fclose(FILE* fid);

void* malloc(int tam);
void free(void* pt);
```

Documentación de sockets:

```
SOCKET sock;

SOCKET socket(int address_family,int sock_type,SOCK_STREAM,int protocol);
// Usar address_family=AF_INET, sock_type = SOCK_DGRAM o SOCK_STREAM, protocol=0

int bind(SOCKET sock,struct sockaddr_in *addr_local,int tam_sockaddr_in);
SOCKET connect(SOCKET sock, struct sockaddr_in *addr_remoto,int tam_sockaddr_in);
int listen(SOCKET sock,int maximo);
int accept(SOCKET sock, struct sockaddr_in *addr_remoto,int tam_sockaddr_in);
int send(SOCKET sock,const char* datos,int nbytes,int flags); // Usar flags=0
int recv(SOCKET sock,char* datos,int nbytes,int flags); // Usar flags=0
int closesocket(SOCKET sock);


struct sockaddr_in {
        short   sin_family;
        u_short sin_port;
        struct  in_addr sin_addr;
        char    sin_zero[8];
};

Para dirección IP, usar:
       addr_laquesea.sin_addr.S_un.S_addr=inet_addr(cad. caract. dirección ip);

Para el Puerto, usar:
       addr_laquesea.sin_port=htons(entero n° de puerto);
```

Control de Procesos en Tiempo Real

Examen Extraordinario - Junio 2012

Documentación de funciones y estructuras POSIX utilizables:

```
int pthread_create(pthread_t* id,pthread_attr* attr,void *(*fn)(void*),void* arg);
int pthread_cancel(pthread_t thread);
void* pthread_join(pthread_t thread);

int sem_init(sem_t *sem, int pshared, unsigned int value); // Usar pshared=0
int sem_wait(sem_t* sem);
int sem_post(sem_t* sem);
int sem_destroy(sem_t* sem);

int pthread_mutex_init(pthread_mutex_t* mutex, const pthread_mutexattr_t *attr);
int pthread_mutex_lock(pthread_mutex_t* mutex);
int pthread_mutex_unlock(pthread_mutex_t* mutex);
int pthread_mutex_destroy(pthread_mutex_t* mutex);

int pthread_cond_init(pthread_cond_t* cond, const pthread_condattr_t *attr);
int pthread_cond_wait(pthread_cond_t* cond,pthread_mutex_t* mutex);
int pthread_cond_broadcast(pthread_cond_t* cond);
int pthread_cond_destroy(pthread_cond_t* cond);

int sigemptyset(sigset_t* set);
int sigaddset(sigset_t* set,int signo);
int pthread_sigmask(int how, const sigset_t *set, sigset_t *oset);
                    /*     how=SIG_BLOCK, SIG_SETMASK ó SIG_UNBLOCK */
int sigaction(int sig, const struct sigaction *act,struct sigaction *oact);
int sigwait(sigset_t* set,int* sig);

struct sigaction
{
       void (*sa_sigaction)(int signo, siginfo_t *info, void *dontuse);
       sigset_t sa_mask;
       int sa_flags; /* Poner bit SA_SIGINFO */
};

int timer_create(clockid_t clockid,struct sigevent* sig,timer_t* id);
                                 // Usar clockid= CLOCK_MONOTONIC y sig=NULL
int timer_settime(timer_t id,int flags,const struct itimerspec* ival,
                  struct itimerspec* oval);   // Usar flags=0, oval=NULL
int timer_delete(timer_t id);

struct timespec
{
       time_t tv_sec;      // Segundos
       long tv_nsec;// Nanosegundos
} ;

struct itimerspec
{
       struct timespec it_interval;     // Periodo de temporización
       struct timespec it_value;        // Primera activación
} ;

mdq_t mq_open(char* nombre,int apert,mode_t modo,mq_attr* attr);
       // 'apert' puede ser combinación con | de: O_CREAT, O_WRONLY, O_RDONLY. Si
       'apert' contiene O_CREAT, hay que añadir los parámetros modo=0 y attr, sino no
       se ponen.

mdq_t mq_open(char* nombre,int apert,mode_t modo,mq_attr* attr);
int mq_send(mdq_t idcola,char* datos,int nbytes,int prioridad);
int mq_receive(mdq_t idcola,char* datos,int nbytes,int* prioridad);
int mq_unlink(mdq_t idcola);

typedef struct mq_attr {
       long mq_flags; /*usar 0 */
       long mq_maxmsg; /*número máximo de mensajes*/
       long mq_msgsize; /*tamaño máximo del mensaje*/
       long mq_curmsgs; /*actual número de mensajes*/
} mq_attr_t;
```