Control de Procesos en Tiempo Real

Examen Ordinario - Febrero 2007

---

1) *3 puntos*

Realizar un programa que, dado un nº entero 'n' y un nombre de archivo pasados en línea de comandos, lea del archivo (formato texto) una tabla de 'n' datos de tipo real, y genere con los mismos una matriz de Vandermonde de orden 'n':

$$\text{tabla} = [a_0 \quad a_1 \quad a_2 \quad a_3 \ldots \quad a_{n-1}]$$

$$\text{matriz} = \begin{bmatrix} 1 & 1 & 1 & \ldots & 1 \\ a_0 & a_1 & a_2 & \ldots & a_{n-1} \\ a_0^2 & a_1^2 & a_2^2 & \ldots & a_{n-1}^2 \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ a_0^{n-1} & a_1^{n-1} & a_2^{n-1} & \ldots & a_{n-1}^{n-1} \end{bmatrix}$$

Dicha matriz será pasada a una función Determinante() que devuelve un valor real que hay que escribir en pantalla. No hay que hacer la función Determinante(), aunque sí escribir su prototipo y la llamada a la misma.

NOTA: Si no se saben pasar los parámetros en línea de comandos, pedirlos por pantalla.

NOTA: No se puede suponer un tamaño máximo para 'n'.

2) *3 puntos*

En un Sistema de T.R. se desea crear un hilo que, cada "Tm" ms, lea un nuevo valor en un canal A/D, y añada a una lista enlazada un valor que se corresponda con el resultado de la media de los 5 últimos valores (en Voltios) leídos del canal A/D. Para ello, se dispone en la librería "ConvAD.lib" de la función siguiente, cuyo prototipo está declarados en "ConvAD.h" (ambos archivos se encuentran en el directorio C:\EXAMEN):

```
int LeeAD();     // Lee un valor del conversor A/D. El valor devuelto
                 // corresponde a la tensión en el canal A/D,
                 // según la relación lineal: 0 (leido) = min = 0V,
                                              1023 (leido) = max = 5V)
```

Condiciones:

- El hilo recibe como parámetro en la inicialización (ver doc. en última página), el periodo de conversión Tm.
- Se debe activar periódicamente la lectura A/D mediante una temporización cada Tm milisegundos.
- La lista debe estar accesible para su utilización por otros hilos, que pueden retirar los primeros elementos de la lista. Por ello, se debe proteger el acceso compartido a la misma.
- Escribir la parte de código que lanzaría a este hilo.
- Indicar los pasos que se realizarían en Visual C++ para incluir en el proyecto la librería necesaria, y poder compilar y enlazar sin errores.
- ¿La temporización utilizada ha sido síncrona o asíncrona? Explicar la diferencia entre ambas.

Control de Procesos en Tiempo Real

Examen Ordinario - Febrero 2007

---

3) *2 puntos*

Dos tareas de un Sistema de Tiempo Real deben compartir una tabla de 10 datos enteros, con el mecanismo productor/consumidor. ¿Qué mecanismo de sincronización utilizaría para las mismas en los casos siguientes?

a) Ambas tareas se implementan como hilos de un mismo proceso y se desea un mecanismo de sincronización rápido y sencillo.

b) Ambas tareas se encuentran en procesos distintos, y se puede dar el caso de que la tarea productora genere varias tablas de datos antes de que la consumidora las utilice. No se desea perder ningún valor intermedio.

c) Ambas tareas se ejecutan en procesos que pueden estar en equipos distintos.

Implemente en código C un ejemplo de envío de los datos entre los 2 equipos para uno de los casos anteriores (a elegir).

4) *2 puntos*

Contestar **razonadamente** a las siguientes preguntas:

a) Si se utilizara un RTOS para el sistema de tiempo real de complejidad media-baja. ¿qué ventajas e inconvenientes tendría su uso?

b) ¿Qué combinación hardware/software debería ser utilizadas, si el factor coste por unidad no es muy importante, para un sistema de T.R. con necesidades de procesamiento elevadas y que debe adaptarse fácilmente a cambios y novedades en dispositivos de E/S? Razonar la respuesta.

- DSP con Sistema Operativo de Tiempo Real.
- Microcontrolador con Sistema Operativo de Tiempo Reall.
- PC industrial con Sistema Operativo de Tiempo Reall
- Microcontrolador con selección de tareas por Function Queue Scheduling.

c) ¿Por qué se deben utilizar funciones específicas y no las funciones genéricas fopen(), fscanf(), fprintf() etc. para la lectura y escritura de canales A/D y D/A?

Control de Procesos en Tiempo Real

Examen Ordinario - Febrero 2007

Pase 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.
```

Conversión de cadena de caracteres a entero:

```
int atoi(char* cadena);
```

Documentación de funciones y estructuras POSIX a utilizar:

```
int pthread_create(pthread_t* id,pthread_attr* attr,void *(*fn)(void*),void* arg);
                   // arg es un puntero al parámetro que se quiere pasar al hilo
                   // usar attr=NULL

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);
                                          // Usar attr=NULL
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);
                                          // Usar attr=NULL
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 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 sigwait(sigset_t* set,int* sig);   // usar *set=SIGRTALL para el timer
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
} ;
```

Control de Procesos en Tiempo Real

Examen Ordinario - Febrero 2007

```
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;
```

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);
int 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,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("direccion ip");
```