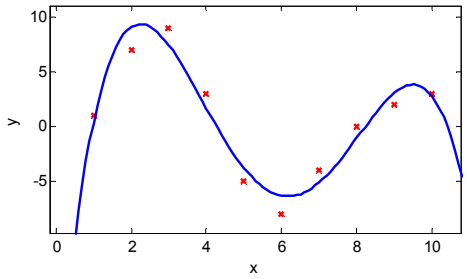
Control de Procesos en Tiempo Real

Examen Ordinario - Febrero 2008

---

1) *3 puntos*

Se desea realizar **una función** que calcule por el método de mínimos cuadrados el polinomio de orden 'n' que mejor ajusta a un conjunto de 'm' datos X/Y.



Dado un conjunto de 'm' parejas x_i/y_i, se puede calcular el polinomio de ajuste $y = a_0 + a_1.x + a_2.x^2 + ... + a_n.x^n$ de la forma siguiente:

$$\begin{bmatrix} 1 & x_0 & x_0^2 & ... & x_0^n \\ 1 & x_1 & x_1^2 & ... & x_1^n \\ 1 & x_2 & x_2^2 & ... & x_2^n \\ ... & ... & ... & ... & ... \\ 1 & x_{m-1} & x_{m-1}^2 & ... & x_{m-1}^n \end{bmatrix} \bullet \begin{bmatrix} a_0 \\ a_1 \\ ... \\ a_n \end{bmatrix} \approx \begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ ... \\ y_{m-1} \end{bmatrix}$$

$$M_x \cdot A \approx M_y \;\mapsto A = \left(M_x^t.M_x\right)^{-1} \cdot M_x^t.M_y$$

Para la realización de esta función, se dispone de una librería de cálculo matemático que permite asignación y liberación de memoria, y el cálculo de transpuesta, producto e inversa de matrices dinámicas. Esta librería está compuesta por el archivo de cabecera "matrices.h" y la librería propiamente dicha "matrices.lib", que se encuentran en el directorio "C:\Examen\CPTR\matrices".

- Escribir un posible archivo de cabecera "matrices.h".
- Escribir el código de la función AjustePolinomio (compatible con el anterior).
- Indicar los pasos que se realizarían en Visual C++ para incluir en el proyecto la librería necesaria, y poder compilar y enlazar sin errores.

---

2) *3 puntos*

En un Sistema de T.R. se desea crear un hilo que, cada 3250 ms, lea la primera línea de un archivo de texto y compruebe si en la misma se encuentra la orden "SUMAR=valor" y, en tal caso, se añada a la variable 'suma' el valor indicado (enteros). El hilo debe recibir como parámetros en su inicialización (ver doc. en última página) las direcciones donde se encuentran la variable 'suma' y el nombre del archivo. El hilo termina si la línea del archivo contiene exactamente la orden "FINAL".

- Escribir el código de este hilo. No usar sleeps para la temporización.
- Escribir la parte de código que lanzaría a este hilo.
- ¿Se ha protegido el posible acceso compartido al archivo? ¿Y a la variable suma? Justificar ambas respuestas.
- ¿La temporización utilizada ha sido síncrona o asíncrona? Explicar la diferencia.

Control de Procesos en Tiempo Real

Examen Ordinario - Febrero 2008

---

3) Un Sistema de Tiempo Real tiene dos tareas.

2 puntos

La tarea T1 debe realizar una adquisición de datos, procesarlos y, si el resultado cumple ciertas condiciones, avisar a la tarea T2 para que realice una serie de acciones con los datos procesados.

La tarea T2 sólo tiene trabajo cuando la tarea T1 le avisa; en este caso, comprueba los resultados de T1, pone al estado adecuado ciertas salidas digitales y vuelve a esperar al aviso.

No se considera posible y/o importante para el funcionamiento que la tarea T1 genere un 2º aviso antes de que la tarea T2 haya terminado su procesamiento.

¿Qué mecanismo de sincronización utilizaría para las mismas en los casos siguientes? (Justificar las respuestas)

a) Ambas tareas se implementan como hilos de un mismo proceso.

b) Se pasa a considerar posible e importante que la tarea T1 genere varios avisos antes de que la tarea T2 termine, y no se desea perder ninguno de ellos.

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

Escriba un código de ejemplo para cada caso (no la parte de inicialización del mecanismo de sincronización, sólo ejemplos de las llamadas a las funciones de sincronización).

4) Cuestiones:

1 punto

a) ¿Qué es un lenguaje de programación seguro?

b) ¿Es C un lenguaje de programación seguro?

c) ¿Qué ventajas y qué inconvenientes aporta un lenguaje seguro en la programación de Sistemas de Tiempo Real?

5) Cuestiones:

1 punto

a) ¿Qué es una función reentrante?

b) Poner un ejemplo de función que haga la suma de una tabla de forma reentrante y no-reentrante.

c) ¿En qué casos hay que tener en cuenta si una función es o no reentrante?

Control de Procesos en Tiempo Real

Examen Ordinario - Febrero 2008

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");
```