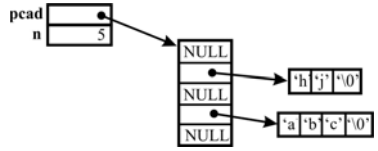
Examen de Sistemas Informáticos en Tiempo Real
IAEI (18/12/07)

CUESTIONES. Tiempo: 1 hora 15 minutos (Valoración: 50%).

Advertencia: Se piden sobre todo **conceptos**, más que ejemplos concretos de programación, salvo que se pida expresamente.

1. Explique para qué se utilizan el puntero de pila y la estructura de datos que se relaciona directamente con él.
2. Programe en C la función **mayor**, que recibe como argumento un puntero a una estructura de tipo **tcad** y devuelve un valor entero. En la estructura de tipo **tcad** el primer elemento, **pcad**, es un puntero que define una tabla de punteros a **char**; la dimensión de la tabla la define el segundo elemento, **n**. Cada puntero define una cadena de caracteres si no vale NULL. La función **mayor** devolverá el número de caracteres de la cadena más larga, sin contar terminadores (un puntero NULL equivale a una cadena de longitud cero); en el ejemplo de la figura, el resultado de la función sería 3. **No utilice funciones de biblioteca**. Es necesario acompañar el código de comentarios o explicaciones.

```
struct tcad
{
   char **pcad;
   int n;
}
```



3. Realice un ejecutivo cíclico con las actividades que figuran en la tabla, explicando el diseño. Diga cuáles son los ciclos principal y secundario, así como la distribución temporal de actividades, demostrando que se cumplen las restricciones temporales. Para todas las actividades el tiempo límite ("deadline") es igual al periodo.

Nombre	Periodo (ms)	Coste (ms)
A	30	6
B	20	2
C	60	15

4. Para el siguiente código a) Explique **en general** su funcionamiento, suponiendo que el proceso que se crea al ejecutarlo recibe solamente señales SIGRTMIN acompañadas de un dato b) Compruebe si el funcionamiento es predecible para cualquier valor del dato que acompaña a la señal, y si no lo es, explique qué modificaciones haría para que lo fuera.

```
<cabeceras correctas>
void f(int n, siginfo_t *n1, void *aux) { }
int main(int a, char **a1) {
    int i, j, k; sigset_t s1;
    struct sigaction sa; siginfo_t v;
    sigemptyset(&s1);
    sigaddset(&s1, SIGRTMIN);
    sa.sa_flags = SA_SIGINFO;
    sa.sa_sigaction = f;
    sigfillset(&sa.sa_mask);
    sigprocmask(SIG_BLOCK, &s1,NULL);
    do {
        sigwaitinfo(&s1, &v);
        j = v.si_value.sival_int;
        if(!fork()) {
            execl(a1[j], a1[j], NULL);
            printf("Mensaje 1\n");
            exit(1);
        }
        wait(&k);
    } while(!WEXITSTATUS(k));
    return 0;
}
```

5. Explique los siguientes conceptos relativos a las colas de mensajes de POSIX 10031b:
   - Creación y apertura de una cola, con garantía de que sea de nueva creación (llamadas que se utilizan, significado y uso de los argumentos).
   - Cierre y destrucción de una cola de mensajes (llamadas que se utilizan y características de tales operaciones).

Datos que pueden ser útiles:

```
int sigaction(int sig, struct sigaction *act, struct sigaction *oact);
int kill(pid_t pid, int sig);
int sigwaitinfo(const sigset_t *estas_sg, siginfo_t *infop);
void exit(int status);
int mq_send(mqd_t cola, const char *datos, size_t longitud, unsigned int prioridad);
int mq_receive(mqd_t cola, const char *datos, size_t longitud, unsigned int *prioridad);
int sigemptyset(sigset_t *pset); int sigfillset(sigset_t *pset);
void *malloc(size_t tam); void free(void *p);
int sigaddset(sigset_t *pset, int sig); int sigdelset(sigset_t *pset, int sig);
int sigprocmask(int how, const sigset_t *set, sigset *oset);
int sigqueue(pid_t pid, int sig, const union sigval val);
int mq_notify(mqd_t cola, const struct sigevent *espec); pid_t fork(void);
int pthread_cond_signal(pthread_cond_t *cond);
int pthread_cond_wait(pthread_cond_t *cond, pthread_mutex_t *mutex);
pid_t getpid(void); pid_t getppid(void);
int mq_getattr(mqd_t cola, struct mq_attr *atributos);
int mq_close(mqd_t cola); int mq_unlink(const char *nombre);
int mq_notify(mqd_t cola, const struct sigevent *espec);
struct mq_attr {    long mq_maxmsg;
                    long mq_msgsize;
                    long mq_flags;
                    long mq_curmsgs; };
int timer_create(clockid_t reloj, struct sigevent *aviso, timer_t *tempo);
mqd_t mq_open(const char *mq_name, int oflag, mode_t modo, struct mq_attr *atributos);
int timer_settime(timer_t tempo, int flags, const struct itimerspec *spec, struct itimerspec *spec_ant);
Modo: S_I + (R, W, X) + (USR, GRP, OTH). También S_IRWXU, S_IRWXG, S_IRWXO
Flags: O_RDONLY, O_WRONLY, O_RDWR, O_CREAT, O_EXCL, O_APPEND, O_TRUNC,
O_NONBLOCK
```

Examen de Sistemas Informáticos en Tiempo Real
IAEI (18/12/07)

PROBLEMA. TIEMPO: 1 HORA 30 MINUTOS (VALORACIÓN: 50%).

Resumen: Se pide realizar en C y con librerías POSIX un programa para crear un proceso **multihilo** que controla el sistema de la figura:

- El **vehículo 1** transporta una pieza de la cinta 1 (izquierda) y las descarga en las posiciones de almacenamiento (centro).
- El **vehículo 2** carga piezas en las posiciones de almacenamiento y las descarga en la cinta 2 (derecha)**.**
- Cuando cuando todas las posiciones de almacenamiento están vacías durante más de 5 minutos seguidos **se genera una alarma**.

Sensores y actuadores:

Salidas: M1 (mover la cinta 1), CV1 y CV2 (descargar pieza del vehículo 1 o 2), VV1 y VV2 (cargar pieza en vehículo 1 o 2). La función **void cambia_salida(int out, int valor)** da el valor **valor** a la salida **out**.

Entradas: P1 y P2 (detectores de presencia). El cambio de valor de P1 y P2 (de cero a uno **o** de uno a cero) provoca el envío de una **señal POSIX** al proceso, con un dato que indica el nuevo valor de la entrada (SIGRTMIN para P1 y SIGRTMIN + 1 para P2).

Movimiento de los vehículos: Para controlar los vehículos se dispone de la función **void mueve(int veh, float fila, float col)**, que mueve el vehículo **veh** (1 o 2) a la posición **fila**, **col**. Se conocen las posiciones que se muestran en la figura. Hay **NPA** posiciones de almacenamiento, con coordenadas separadas 1 unidad de longitud en la dirección de las columnas (horizontal en la figura).

Los códigos numéricos de los actuadores y las posiciones están definidos como constanes simbólicas en la cabecera **sistema.h** con el mismo nombre que figura en este enunciado. La funciones **cambia_salida** y **mueve** están disponibles en una biblioteca. Puede suponerse que inicialmente no hay piezas en ninguna parte y los vehículos están al lado de las cintas.

Funcionamiento de los vehículos:

Vehículo 1: Arranca la cinta para obtener la pieza y la carga con CV1. A continuación espera a que alguna posición de almacenamiento esté libre y accesible. Cuando esto sucede mueve el vehículo hasta ella y descarga la pieza con VV1. Finalmente vuelve a la cinta 1 a por otra pieza.

Vehículo 2: Espera a que alguna posición de almacenamiento esté ocupada y sea accesible. Cuando esto sucede mueve el vehículo hasta ella, carga la pieza con CV2, vuelve a la cinta 2 y la descarga con VV2, pero primro espera a que P2 esté desactivado (a que no haya piezas en P2). La cinta 2 no es controlada por el proceso.

Los dos vehículos nunca deben entrar a la vez en la zona que está delante de las posiciones de almacenamiento. Los ciclos se repiten hasta que se para el sistema.

Envio de estado de almacén y generación de alarma:

Si el almacén **está vacío durante 5 o más minutos seguidos**, el proceso generará una SIGRTMIN cada 200 ms al proceso padre hasta que la condición desaparezca.

Parada:

Cuando recibe la señal SIGTERM o la señal SIGINT el proceso acaba, pero antes los vehículos deben de completar su última operación de carga o descarga en las cintas.

NOTAS:

- Inicialmente puede considerarse que todas las posiciones de almacenamiento están vacías.
- Las operaciones deben ser tan concurrentes (simultáneas) como sea posible.
- Se recomienda tratar las señales sincronamente.
- Deben utilizarse **temporizadores POSIX**, no retrasos.
- **Utilice mutex y variables de condición** para **sincronizar el acceso a las variables compartidas,** evitando (cuando sea posible) permanecer un tiempo largo en la sección crítica.
- Las constantes y prototipos de funciones se encuentran en la cabecera **sistema.h**.
- No es necesario considerar tratamiento de errores en las llamadas al sistema.
- Es preciso acompañar el programa de pseudocódigo o explicación de su funcionamiento.

```
/* Cabecera sistema.h */

/* Constantes simbólicas */
/* Entradas */
#define SP1     SIGRTMIN
#define SP2     SIGRTMIN+1

/* Salidas */
#define M1      0
#define CV1     1
#define CV2     2
#define VV1     3
#define VV2     4

/* Posiciones */
#define FP      10 /* Fila comun */
#define RV1     1  /* Reposo veh 1 */
#define PA0     3  /* Pos. almac. 0 */
#define RV2     20 /* Reposo veh 2 */

/* Numero de posiciones de almac. */
#define NPA     6

/* Funciones disponibles en biblioteca */
void cambia_salida(int out, int valor);
void mueve(int veh, float fila, float col);
```