HOJA DE REFERENCIA – API DE NSYSTEM
NIVEL USUARIO

Listado de referencia de prototipos de funciones para nSystem 2005 (nSystem.h)

- **La tarea principal provista por el programador**

  - int nMain(/* int argc, char **argv */);
    No siempre será necesario colocar argc y argv, por ello los argumentos están comentados

---

- **Creación y manejo de tareas**

  - nEmitTask(nProc, ...);
    Crea una nueva tarea

  - void nExitTask(int rc);
    Termina la tarea que la invoca

  - int nWaitTask(nTask task);
    Espera el término de otra tarea

  - void nExitSystem(int rc);
    Termina todas las tareas (shutdown del proceso Unix)

- **Definición de parámetros para las tareas**

  - int nSetStackSize(int size);
    Tamaño de stack para las nuevas tareas

  - void nSetTimeSlice(int slice);
    Tamaño de la tajada de tiempo (en ms)

  - void nSetTaskName(char *format, ...);
    Nombre de la tarea. Util para debugging

  - void nSetPriority(int pri);
    Prioridad de la tarea actual

  - nTask nCurrentTask();
    El identificador de la tarea actual

  - char* nGetTaskName();
    El nombre de esta tarea

  - int nGetContextSwitches();
    Obtiene el n° de cambios de contexto para la tarea actual

  - int nGetQueueLength();
    Obtiene el largo de la cola

- **Colas FifoQueue**

  - FifoQueue MakeFifoQueue();
    El constructor

  - void PutObj(FifoQueue q, void* o);
    Agrega un objeto al final

  - void* GetObj(FifoQueue q);
    Retorna y extrae el primer objeto

  - int EmptyFifoQueue(FifoQueue q);
    Verdadero si la cola esta vacia

  - void DestroyFifoQueue(FifoQueue q);
    Elimina la cola

  Procedimientos adicionales

  - int LengthFifoQueue(FifoQueue q);
    Entrega el largo de la cola

  - int QueryObj(FifoQueue q, void* o);
    Verdadero si o esta en la cola

  - void DeleteObj(FifoQueue q, void* o);
    Elimina o si esta en la cola

  - void PushObj(FifoQueue q, void* o);
    Agrega un objeto al principio

- **Mensajes**

  - int nSend(nTask task, void *msg);
    Envía un mensaje a una tarea

  - void *nReceive(nTask *ptask, int max_delay);
    Recepción de un mensaje

  - void nReply(nTask task, int rc);
    Responde un mensaje

  - void nSleep(int delay);
    Suspende el proceso por delay milisegundos

  - int nGetTime();
    Entre la hora en milisegundos y módulo “maxint”

- **Semáforos**

  - nSem nMakeSem(int count);
    Construye un semáforo

  - void nWaitSem(nSem sem);
    Operación Wait

  - void nSignalSem(nSem sem);
    Operación Signal

  - void nDestroySem(nSem sem);
    Destruye un semáforo

- **Monitores**

  - nMonitor nMakeMonitor();
    Construye un monitor

  - void nDestroyMonitor(nMonitor mon);
    Destruye un monitor

  - void nEnterMonitor(nMonitor mon);
    Ingreso al monitor

  - void nExitMonitor(nMonitor mon);
    Salida del monitor

  - nCondition nMakeCondition(nMonitor mon);
    Construye una condición

  - void nDestroyCondition(nCondition cond);
    Destruye una condición

  - void nWaitCondition(nCondition cond);
    Operación Wait

  - void nSignalCondition(nCondition cond);
    Operación Signal

- **E/S básica**

Estas funciones son equivalentes a open, close, read y write en Unix. Las “nano” funciones son no bloqueantes para el proceso Unix, solo bloquean la tarea que las invoca.

  - int nOpen(char *path, int flags, ...);
    Abre un archivo

  - int nClose(int fd);
    Cierra un archivo

  - int nRead(int fd, char *buf, int nbyte);
    Lee de un archivo

  - int nWrite(int fd, char *buf, int nbyte);
    Escribe en un archivo

Estas funciones se pueden usar en caso de necesitar que la E/S estándar sea no bloqueante (ver examples1/iotest.c). En algunos casos como curses, es inevitable que sea bloqueante.

- void nReopenStdio();
  Reabre la E/S estándar

- void nSetNonBlockingStdio();
  Coloca en modo no bloqueante la E/S std.

- **Los servicios**

  - int nFprintf(int fd, char *format, ...);
    Permite escribir en el descriptor spedificado e fd

  - int nPrintf(char *format, ...);
    Permite escribir a pantalla

  - void nFatalError(char *procname, char *format, ...);
    Imprime un mensaje con el código del error

  - void *nMalloc(int size);
    Obtener un puntero a un bloque de memoria (igual a malloc)

  - void nFree(void *ptr);
    Libera la memoria apuntada por ptr

  - int nGetTime();
    Obtiene el tiempo actual del sistema