INDICE

TEMA 1 : BIBLIOTECA ESTÁNDAR

stdio.h

Contiene definiciones de macros, prototipos de funciones y declaraciones de variables que reconocen el entorno.

Definiciones de tipos :

typedef unsigned int **size_t**

typedef long **fpos_t**

typedef long **wchar_t**

typedef long **wint_t**

Definiciones de macros :

#define **NULL** 0 Para punteros

#define **EOF** −1 Devuelto al leer más allá del final del archivo

#define **SEEK_SET** 0 Principio de archivo

#define **SEEK_CUR** 1 Posición actual

#define **SEEK_END** 2 Fin de archivo

#define **stdin** (&_iob[0]) Entrada estándar

#define **stdout** (&_iob[1]) Salida estándar

#define **stderr** (&_iob[2]) Salida de errores estándar

Definición del tipo FILE :

typedef struct _FILE

{ int **_cnt** ; Número de caracteres disponibles en el buffer

unsigned char ***_ptr** ; Puntero al carácter siguiente

unsigned char ***_base** ; Puntero al buffer que contiene la info. del archivo

unsigned char **_flag** ; Flag de acceso (modo de acceso)

unsigned char **_file** ; Descriptor del archivo

unsigned char **_buf[2]** ;

}FILE ;

Prototipos de funciones para archivos de texto :

int ***fclose** (FILE *fd)

fd : Descriptor de archivo

int **fflush** (FILE *fd)

Vacia el contenido de una secuencia de salida. Esta función escribe todos los datos almacenados en el buffer sobre el archivo asociado con *fd*. Si se llama a *fflush* con un puntero nulo se vacían los buffers de todos los archivos abiertos.

La función *fflush* devuelve 0 si tiene éxito ; en otro caso devuelve EOF.

FILE ***fopen** (const char *Nombre, const char *Modo)

Modo	**Significado**
r	Abre un archivo de texto para lectura
w	Crea un archivo de texto para escritura
a	Abre un archivo de texto para añadir
rb	Abre un archivo binario para lectura
wb	Crea un archivo binario para escritura
ab	Abre un archivo binario para añadir
r+	Abre un archivo de texto para lect/escr

w+	Crea un archivo de texto para lect/escr
a+	Añade o crea un archivo de texto para lect/escr
r+b	Abre un archivo binario para lect/escr
w+b	Crea un archivo binario para lect/escr
a+b	Añadir en un archivo binario en modo lect/escr

Esta función devuelve un puntero a un archivo. En caso de error al intentar abrir el archivo devuelve un puntero nulo.

size_t **fread** (void *buffer, size_t bytes, size_t num, FILE *fd)

Buffer : Es un puntero a una región de memoria (contenedor).

Bytes : Número de bytes a leer.

Num : Es el número de elementos a leer (con *bytes* de longitud)

Fd : Es un descriptor de archivo.

La función *fread* devuelve el número de elementos leídos. Este valor puede ser menor que *cuenta* si se encuentra el final del archivo o si se produce un error.

size_t **fwrite** (void *buffer, size_t bytes, size_t num, FILE *fd)

Buffer : Es un puntero a una región de memoria (contenedor).

Bytes : Número de bytes a escribir.

Num : Es el número de elementos a escribir (con *bytes* de longitud)

Fd : Es un descriptor de archivo.

La función *fwrite* devuelve el número de elementos escritos. Este valor será igual a *cuenta* a menos que se produzca un error.

Prototipos de funciones de tratamientos de error :

int **feof** (FILE *fd)

Devuelve cierto si se alcanzado el final del archivo ; en otro caso, devuelve 0 (para archivos binarios).

int **ferror** (FILE *fd)

Determina si se ha producido un error en una operación sobre un archivo.

Devuelve cierto si se ha producido un error durante la última operación

sobre el archivo, sino devuelve falso.

void **clearerr** (FILE *fd)

Se utiliza para inicializar a 0 el indicador de error del archivo.

Prototipos de funciones de acceso directo :

int **fseek** (FILE *fd, long numbytes, int origen)

Situa el indicador de posición del archivo.

Numbytes : Número de bytes a partir de *origen* que se moverá *fd*.

Origen : Es una de las siguientes macros (SEEK_SET,SEEK_CUR,

SEEK_END) ;

Devuelve 0 si ha tenido éxito y un valor distinto de 0 cuendo hay un error.

void **rewind** (FILE *fd)

Inicializa el indicador de posición, al principio del archivo.

long **ftell** (FILE *fd)

Devuelve el valor actual del indicador de porsición del archivo. Retorna

−1L cuando se produce un error

(Además tenemos stdlib.h y string.h)

TEMA 2 : HERRAMIENTAS DE AYUDA AL DESARROLLO

EN EL ENTRONO UNIX

Compilador cc

Es un guión shell que traduce el código fuente a código objeto y crea un ejecutable. Además enlaza sus módulos.

Prog.c 1ª Fase Prog.i 2ª Fase Prog.s 3ª Fase Prog.o

Fases al ejecutar : $ cc 4ª Fase

Ejecutable

Podemos indicar que se detenga en una fase concreta :

−**P** : Que se detenga en la primera fase (c sin macros)

−**s** : Que se detenga en segunda fase (ensamblador)

−**c** : Que se detenga en la tercera fase

Sin nada llega al ejecutable, que por omisión se llama a.out.

−**o** nombre : Para darle un nombre al ejecutable.

Las aplicaciones se pueden dividir en módulos que son ficheros fuente con funciones de control de flujo de la aplicación.

$ cc m1.c m2.c .....mk.c −o ejec

Más opciones :

$ cc −**D** macro ó $ cc −**D** macro = valor

−**Aa** : Compila en ansi. En caso contrario lo hace en pre ansi.

Constructor make

Genero procesos de compilación de proyectos en c bajo el paradigma de la descomposición funcional.

Fichero make :

ejec : s1.o s2.o

cc −o ejec s1.o s2.o

s1.o : s1.c define.h

cc −c s1.c

s2.o : s2.c define.h

cc −c s2.c

A este fichero se le puede dar cualquier nombre. Si se le llama pepe la ejecución será

$ make pepe y si se le llama makefile será *$ make*.

Bibliotecas de enlace estático *ee* y de enlace dinámico *ed*

Las librerías compartidas permiten que múltiples programas puedan compartir el código de las funciones incluidas en la librería. Este modo de compartir puede estar implementada de forma **estática** (ed) o **dinámica** (ee).

En una **librería estática** las direcciones virtuales de las funciones son fijas y se enlazan con el fichero a.out cuando se realiza el linkado, de manera que estas direcciones no se modifican aunque las funciones sean modificadas. Cuando se realizan cambios en las funciones de la librería es necesario conocer el espacio de direcciones disponible.

Las **librerías dinámicas** no residen en direcciones fijas de memoria y las funciones se enlazan con el fichero a.out en tiempo de ejecución. Las funciones se pueden cargar en cualquier dirección de memoria.

Para crear **librerías estáticas** se crean con el comando **ar** y tienen el nombre genérico *libxxx.a* :

$ cc −c func1.c func2.c func3.c

$ ar −r libmia.a func1.o func2.o func3.o

La librería se podrá enlazar con el ejecutable de la siguiente manera :

$ cc −ldirectorio main.c fich2.c −lmia

Las **librerías dinámicas** tienen el nombre genérico *libxxx.so* :

$ cc −c func1.c func2.c func3.c

$ cc −G libmia.so func1.o func2.o func3.o

El editor de enlace incluirá en el ejecutable con el que se enlace la librería el nombre del objeto compartido asi como toda la información necesaria para gestionar el enlace en tiempo de ejecución.

$ cc −ldirectorio main.c fich2.c −lmia

Es posible incluir librerías estáticas y otras librerías dinámicas en la lista de ficheros especificados para formar la librería dinámica :

$ cc −G libmia.so −ldirectorio func1.0 func2.o func3.o −lcomp

La opción **dn** permite desactivar el enlace con librerías dinámicas. La orden **cc** proporciona otras dos opciones que permiten combinar enlaces con librerías dinámicas y estáticas :

$ cc −L/home/mlibs/ main.c fich1.c fich2.c −Bstatic lmia fich3.c Bdynamic −lcomp

En la línea anterior se enlaza la librería estática *libmia.a* para resolver las referencias externas de *main.c* y *fich1.c* y *fich2.c* ; por otro lado se indica que emplee la librería *libcomp.so* para resolver las referencias externas no resueltas de esos mismos programas y por último que emplee la librería estándar libc.so para resolver las referencias externas aún no resueltas.

Opciones :

r : Crea la biblioteca y guarda dentro los archivos especificados. Si ya existe

amplia y reemplaza.

d : Elimina los archivos de la biblioteca.

t : Lista el contenido del archivo especificado

x : Exporta al sistema de archivos ese miembro de la biblioteca.

dn : Significa no dinámico. Busca bibliotecas de enlace estático.

TEMA 3 : ESTRUCTURA INTERNA DE UNIX

Introducción