Makefile

Explicación del código

Makefile es un código para compilar una biblioteca estática en C, en este caso la que será lllamada **libft.a**. Esta biblioteca contiene implementaciones de varias funciones de manipulación de cadenas y listas enlazadas, muchas de las cuales son versiones personalizadas de las funciones proporcionadas por la biblioteca estándar de C.

Variables

El Makefile define varias variables que se utilizan a lo largo del archivo:

- **NAME**: el nombre de la biblioteca estática que se va a crear, en este caso **libft.a**.
- **SRCS** y **SRCS_B**: listas de archivos fuente (**.c**) que se van a compilar para crear la biblioteca. La variable **SRCS** contiene las implementaciones de las funciones de cadena, mientras que **SRCS_B** contiene las implementaciones de las funciones de lista enlazada (Bonus).
- **OBJS** y **OBJS_B**: listas de archivos objeto (**.o**) que se crearán al compilar los archivos fuente.
- **CC**: el compilador de C que se va a utilizar, en este caso **cc** (que suele ser un enlace simbólico a **gcc**).
- **CFLAGS**: las opciones que se pasarán al compilador de C. En este caso, se utiliza **-Wall -Wextra -Werror**, que habilita advertencias adicionales y convierte las advertencias en errores.

Reglas

El Makefile define varias reglas para especificar cómo construir la biblioteca:

- La regla **all** es la regla predeterminada y crea la biblioteca estática **libft.a** a partir de los archivos objeto en **OBJS**.
- La regla **bonus** crea una versión adicional de la biblioteca que incluye las implementaciones de las funciones de lista enlazada.
- La regla **%.o: %.c** especifica cómo crear un archivo objeto a partir de un archivo fuente. Compila el archivo fuente con el compilador de C y las opciones especificadas en **CFLAGS**, y crea un archivo objeto con el mismo nombre pero con extensión **.o**.
- Las reglas **clean**, **fclean** y **re** son reglas fantasma que se utilizan para limpiar los archivos objeto y la biblioteca estática.

Explicación del código de este Makefile línea a línea

NAME = libft.a

Esta línea define una variable llamada **NAME** y le asigna el valor **libft.a**. Esta variable se utiliza para especificar el nombre del archivo de la biblioteca estática que se va a crear. En este caso, el nombre de la biblioteca será **libft.a**. La extensión **.a** indica que se trata de una biblioteca estática.

SRCS = ft_memset.c ft_bzero.c ft_memcpy.c ft_memmove.c ft_memchr.c...

Esta línea define una variable llamada **SRCS** y le asigna una lista de archivos fuente (**.c**) que se van a compilar para crear la biblioteca. Estos archivos contienen implementaciones de funciones de cadena y memoria, como **ft_memset**, **ft_bzero**, **ft_memcpy**, entre otras. La lista de archivos se separa con espacios y se utiliza para especificar los archivos que se van a compilar.

SRCS_B = ft_lstnew.c ft_lstadd_front.c ft_lstsize.c ft_lstlast.c ft_lstadd_back.c...

Esta línea define una variable llamada **SRCS_B** y le asigna una lista de archivos fuente (**.c**) que se van a compilar para crear la biblioteca. Estos archivos contienen implementaciones de funciones de lista enlazada, como **ft_lstnew**, **ft_lstadd_front**, **ft_lstsize**, entre otras. La lista de archivos se separa con espacios y se utiliza para especificar los archivos que se van a compilar.

OBJS = $(SRCS:.c=.o)

Esta línea define una variable llamada **OBJS** y le asigna una lista de archivos objeto (**.o**) que se van a crear al compilar los archivos fuente en **SRCS**. La sintaxis **$(SRCS:.c=.o)** indica que se va a reemplazar la extensión **.c** por **.o** en cada archivo de la lista **SRCS**.

OBJS_B = $(SRCS_B:.c=.o)

Esta línea define una variable llamada **OBJS_B** y le asigna una lista de archivos objeto (**.o**) que se van a crear al compilar los archivos fuente en **SRCS_B**. La sintaxis **$(SRCS_B:.c=.o)** indica que se va a reemplazar la extensión **.c** por **.o** en cada archivo de la lista **SRCS_B**.

CC = cc

Esta línea define una variable llamada **CC** y le asigna el valor **cc**, que es el nombre del compilador de C que se va a utilizar para compilar los archivos fuente.

CFLAGS = -Wall -Wextra –Werror

Esta línea define una variable llamada **CFLAGS** y le asigna una lista de opciones que se van a pasar al compilador de C. Las opciones **-Wall -Wextra -Werror** habilitan advertencias adicionales y convierten las advertencias en errores.

all: $(NAME)

Esta línea define una regla llamada **all** que depende de la variable **NAME**. La regla **all** es la regla predeterminada y se utiliza para crear la biblioteca estática **libft.a**.

$(NAME): $(OBJS)

Esta línea define una regla que crea la biblioteca estática **libft.a** a partir de los archivos objeto en **OBJS**. La regla utiliza el comando **$(AR) -r $@ $?** para crear la biblioteca.

bonus: $(OBJS_B)

Esta línea define una regla llamada **bonus** que depende de la variable **OBJS_B**. La regla **bonus** se utiliza para crear una versión adicional de la biblioteca que incluye las implementaciones de las funciones de lista enlazada.

%.o: %.c

Esta línea define una regla que especifica cómo crear un archivo objeto (**.o**) a partir de un archivo fuente (**.c**). La regla utiliza el comando **$(CC) -o $@ -c $(CFLAGS) $<** para compilar el archivo fuente.

clean:

Esta línea define una regla llamada **clean** que elimina los archivos objeto en **OBJS** y **OBJS_B**.

fclean: clean

Esta línea define una regla llamada **fclean** que elimina la biblioteca estática **libft.a** y llama a la regla **clean** para eliminar los archivos objeto.

re: fclean all

Esta línea define una regla llamada **re** que llama a la regla **fclean** para eliminar la biblioteca estática y los archivos objeto, y luego llama a la regla **all** para crear la biblioteca estática de nuevo.

.PHONY: all bonus clean fclean re

Esta línea define una lista de reglas fantasma (phony) que se utilizan para limpiar los archivos objeto y la biblioteca estática. Las reglas fantasma no crean archivos reales y se utilizan solo para especificar dependencias y comandos.

En resumen, el Makefile define variables y reglas para especificar cómo construir una biblioteca estática en C llamada **libft.a**. La biblioteca contiene implementaciones de varias funciones de cadena y memoria, y opcionalmente, funciones de lista enlazada. El Makefile utiliza variables y reglas para automatizar el proceso de construcción y garantizar la consistencia y reproducibilidad de los resultados.

Funcionamiento

Para construir la biblioteca, ejecute el comando **make**. Esto ejecutará la regla **all** y creará la biblioteca estática **libft.a** a partir de los archivos objeto en **OBJS**.

Si desea crear una versión adicional de la biblioteca que incluya las implementaciones de las funciones de lista enlazada, ejecute el comando **make bonus**.

Para limpiar los archivos objeto y la biblioteca estática, ejecute el comando **make clean** o **make fclean**. El primero eliminará solo los archivos objeto, mientras que el segundo eliminará también la biblioteca estática.

El Makefile está bien organizado y fácil de entender, lo que facilita la modificación y el mantenimiento del código. Además, el uso de variables y reglas hace que el proceso de construcción de la biblioteca sea automático y repetible, lo que garantiza la consistencia y la reproducibilidad de los resultados.

Makefile

English version

This code is a Makefile for building a static library in C using the GNU C Compiler (gcc). A Makefile is a file that contains instructions for the **make** build automation tool to compile and build software.

The name of the library being built is **libft.a**. This library contains implementations of various string and memory functions that are similar to those found in the standard C library, but with additional features or optimizations. These functions are defined in the source files listed in the **SRCS** and **SRCS_B** variables.

Here's a breakdown of the different parts of the Makefile:

- The **NAME** variable specifies the name of the library being built.
- The **SRCS** and **SRCS_B** variables are lists of source files that will be compiled to object files. The **SRCS** variable contains the main set of source files, while **SRCS_B** contains additional source files for bonus features.
- The **OBJS** and **OBJS_B** variables are lists of object files that will be created by compiling the source files.
- The **CC** variable specifies the C compiler to use. In this case, it is set to **cc**, which is a synonym for **gcc**.
- The **CFLAGS** variable specifies compiler flags to use when compiling the source files. In this case, it includes **-Wall**, **-Wextra**, and **-Werror**, which enable additional compiler warnings, treat warnings as errors, and enable extra warnings, respectively.
- The **all** target is the default target that is run when you type **make** without any arguments. It depends on the **libft.a** library, which is built by linking together the object files in **OBJS**.
- The **bonus** target builds the bonus features by linking together the object files in **OBJS_B**.
- The **%.o: %.c** rule specifies how to compile a source file into an object file. It uses the **$(CC)** command to compile the source file with the **$(CFLAGS)** flags, and outputs the object file with the same name as the source file but with a **.o** extension.
- The **clean** target removes all object files.
- The **fclean** target removes both the object files and the library.
- The **re** target first runs **fclean** to remove any existing build artifacts, and then runs **all** to build the library from scratch.
- The **.PHONY** special target tells **make** that the **all**, **bonus**, **clean**, **fclean**, and **re** targets are not actual files, but rather phony targets that represent actions to be taken.

In summary, this Makefile defines a set of rules for building a static library in C using the gcc compiler, with various optimizations and additional features. The library contains a set of string and memory functions that are similar to those found in the standard C library, but with additional features or optimizations. The Makefile also includes targets for cleaning up build artifacts and rebuilding the library from scratch.