Tarea Extraclase

de

Sistema Operativo II

Título: LLamada al Sistema en MInix 3.

Integrante:

* Luis Ernesto Glez. Cárdenas.

Introducción

En Minix existen varias llamadas para manejar procesos, surge la problemática de cómo cambiarle el pid a un proceso con el objetivo de poder tratar los procesos por su pid mediante la implementación de una llamada al sistema. Por lo que a lo largo del trabajo explicaremos como desarrollar una llamada al sistema que cumpla con esta condición.

Desarrollo

En Minix 3, los servidores controlan las llamadas al sistema (excepto las muy básicas que están en el microkernel y reciben la denominación especial de tareas del sistema).

Para agregar una nueva llamada al sistema, al nivel de la capa 3, se tienen que realizar dos pasos:

1. Escribir un manipulador (handler) para la llamada al sistema. El manipulador es la función que se invoca en respuesta a una petición de usuario. Cada llamada al sistema tiene su propio manipulador.
2. Escribir una biblioteca de usuario, la cual empaqueta los parámetros de la llamada al sistema e invoca al manipulador sobre el servidor apropiado. Los usuarios siempre invocan las llamadas al sistema a traves de esta biblioteca.

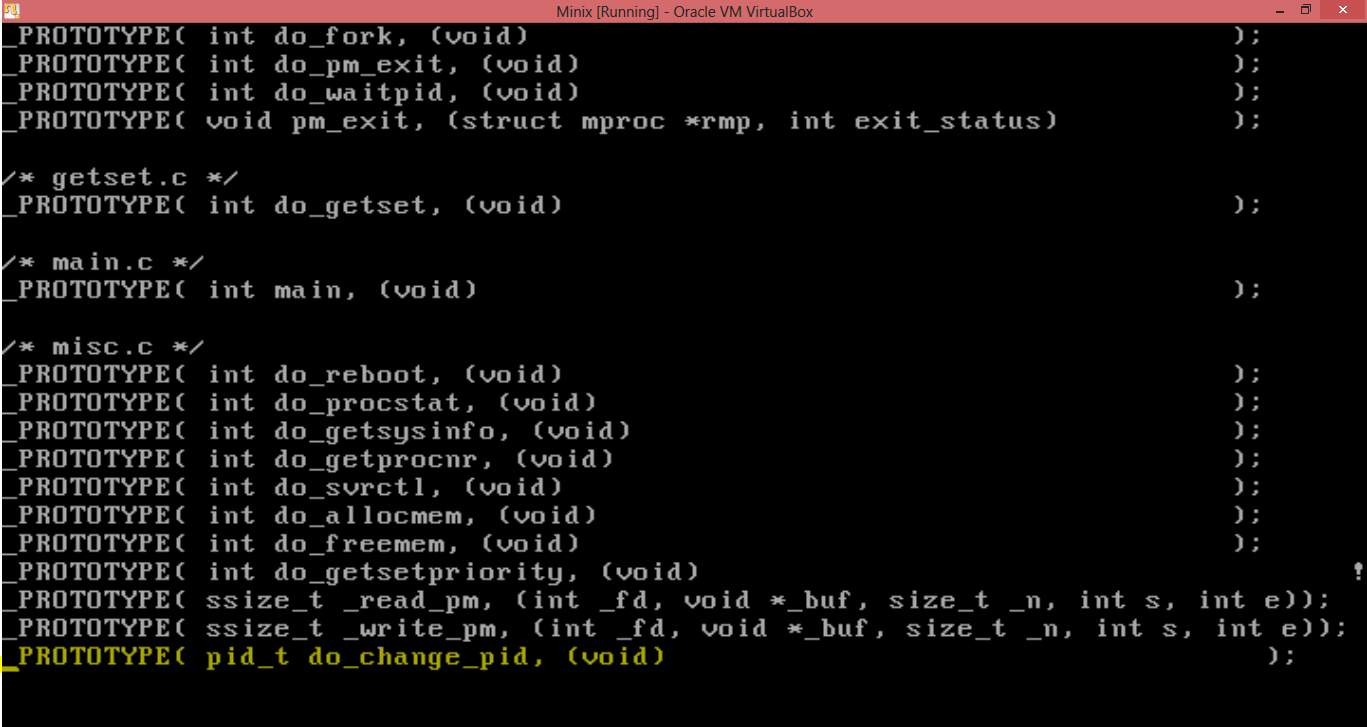
En este caso agregaremos una llamada al sistema en el servidor PM que nos permitirá cambiar el pid a un proceso dado, la cual tiene en cuenta si ese pid que queremos ponerle no está siendo usado por nadie, ni es uno reservado por el Sistema Operativo. La nueva llamada tendrá el siguiente prototipo:

pid\_t change\_pid(pid\_t pidAct, pid\_t pidChang)

# Creación del manipulador de la llamada al sistema

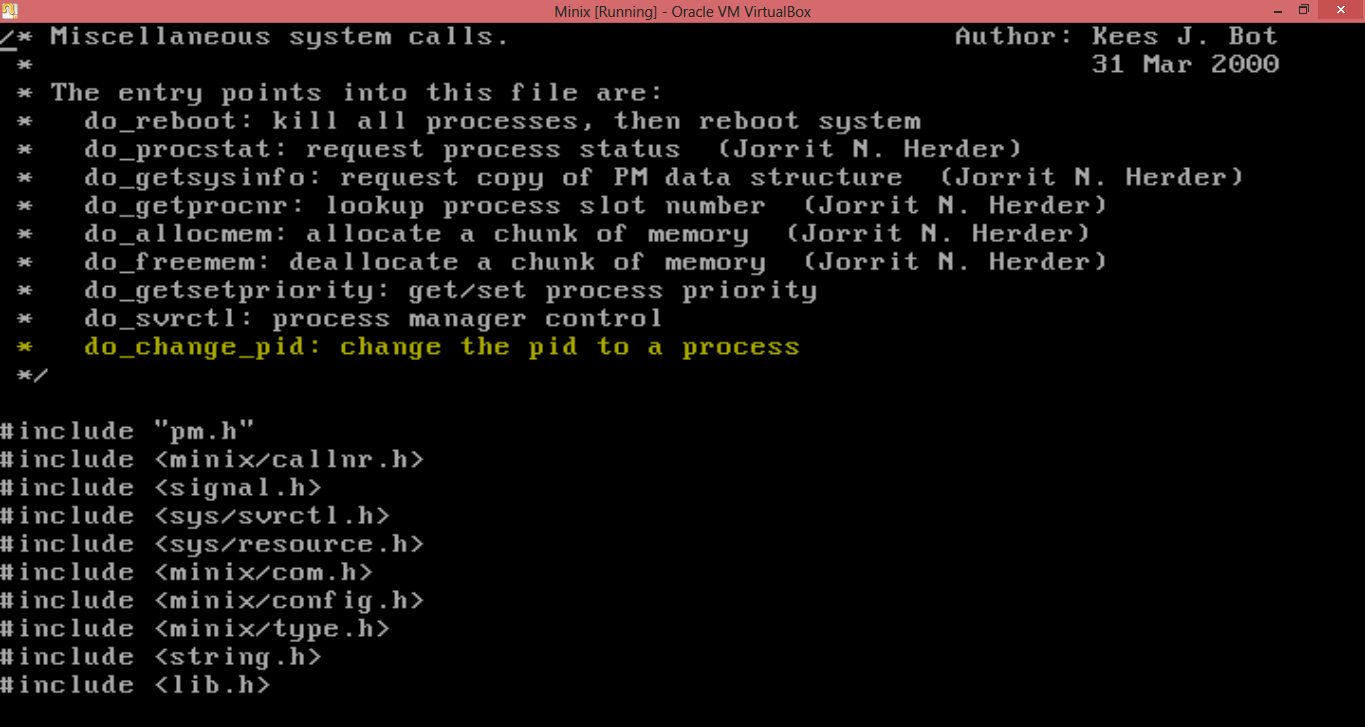
Para crear el manipulador es necesario moverse al directorio **/usr/src/servers/pm**  que contiene los fuentes del servido Manager Process. A continuación editamos el archivo **table.c,** buscando una entrada libre (Fig 1). Este archivo contiene un arreglo de punteros a funciones que es una tabla que se usa para mapear cada *llamada al sistema* con un *número*. En este caso se escogido la entrada 69. Seguidamente pasamos editar el archivo  **proto.h**, que contiene las declaraciones de los prototipos de las funciones (Fig 2). Como la nueva llamada al sistema será agregada en el archivo path.c, entonces su prototipo se pondrá por debajo de la línea de comentario que así lo indica. Esto solo se hace para una mejor organización, pues se puede poner en cualquier posición del fichero. La llamada al sistema devolvería un pid\_t que es un tipo especial de entero y no recibe parámetros pues los necesarios los toma por una estructura llamada message.

Fig(1):table.c

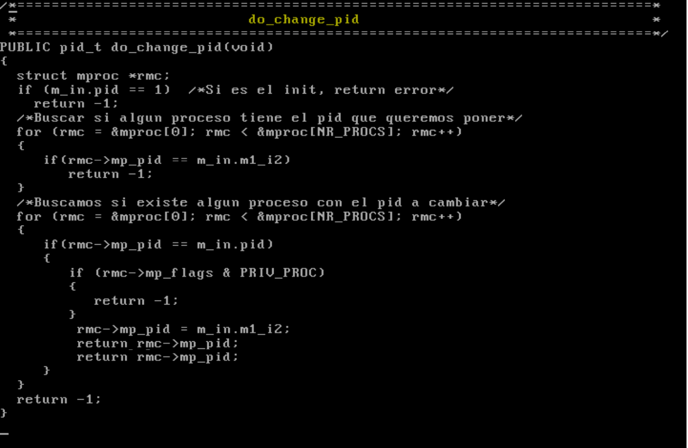


Fig(2):proto.h

Lo próximo sería implementar la llamada al sistema. Para la Implementación se podría crear un fichero aparte para ella sola, pero como se planteó en clases por cuestiones de organización la agregaremos en el fichero misc.c que se encuentra en el mismo directorio. Para seguir el estilo de programación del autor agregamos al inicio del archivo un comentario el cual es el nombre de nuestra llamada y una breve descripción de esta lo cual se muestra en la Fig(3). Luego al final del archivo implementamos la llamada como muestra la Fig(4).



Fig(3):misc.c

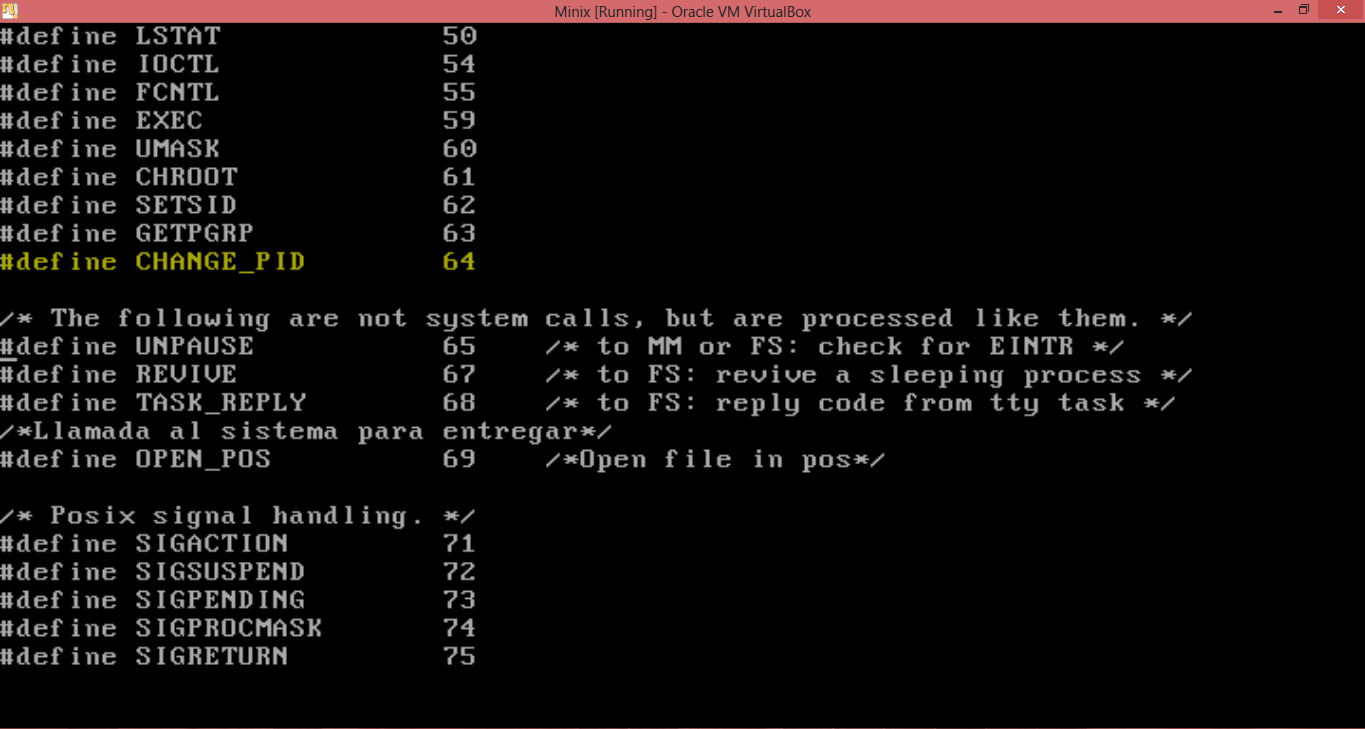


Fig(4):misc.c

Luego de haber hecho todo este proceso es necesario compilar el servidor **pm**, lo cual es muy fácil debido la implementación por módulos que presenta MINIX. Para ello situarse en el directorio del servidor **pm**, ejecutar el comando **make**, el cual analiza las órdenes contenidas en el archivo **Makefile** del mismo directorio para compilar e instalar el nuevo servidor. Por último reiniciar el sistema con el comando **reboot**. para que se use el nuevo servidor.

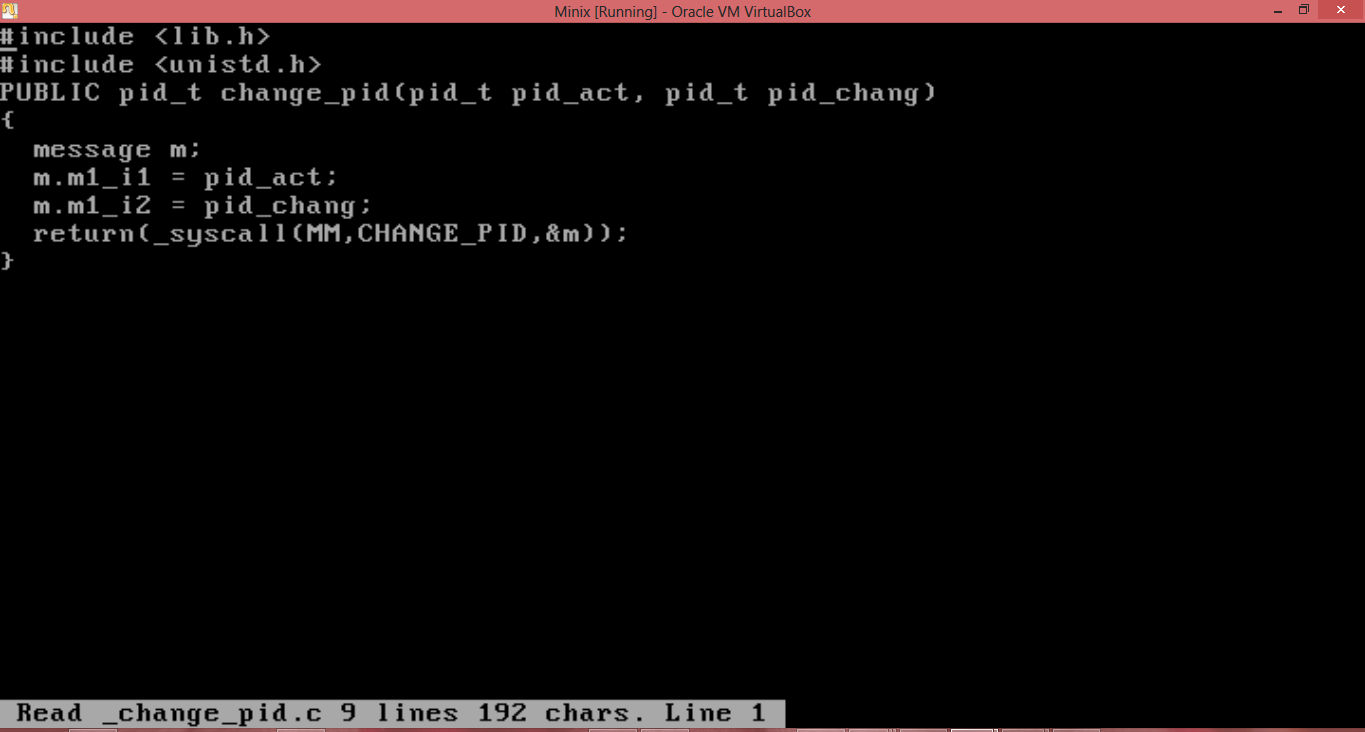
El segundo paso para adicionar una nueva llamada al sistema que quede totalmente funcional, es crear una biblioteca de usuario para que empaquete los parámetros del manipulador de la llamada al sistema en la estructura **message** y se pueda llamar a la función de manera más cómoda. Debe procederse de la forma siguiente:

Editar los archivos **/usr/src/include/minix/callnr.h** y **/usr/include/minix/callnr.h** los cuales se usan con el propósito de mapear las llamadas al sistema con el número que tienen en la tabla se saltos del archivo table.c analizada anteriormente (Fig. 5). Primero se usa **#define** para mapear el número de la llamada al sistema de la función manipuladora con un identificador que en este caso es la entrada 69 del vector definido en table.c.



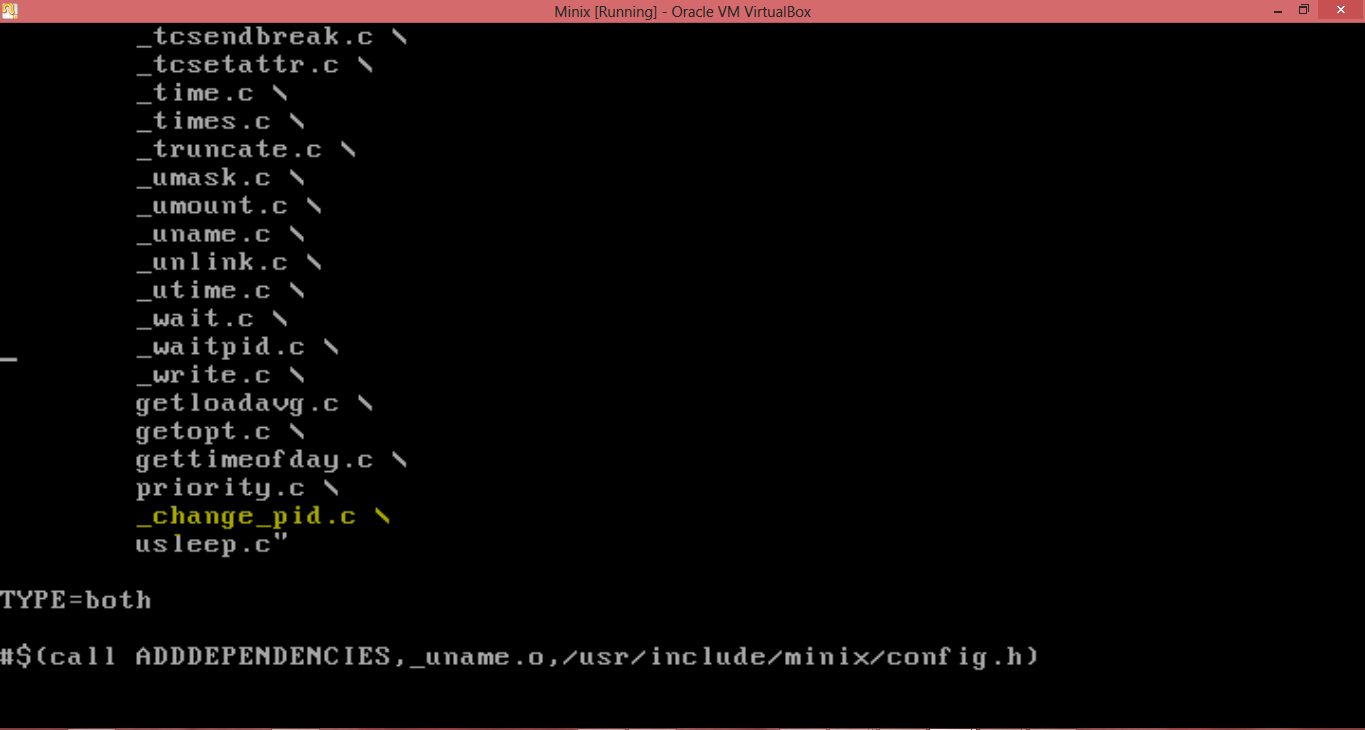
Fig(5):callnr.h

A continuación nos situamos en el directorio de trabajo  **/usr/src/lib/posix.** Aquí se implementa la biblioteca para **do\_change\_pid** en un archivo separado que se denominará **\_change\_pid.c**. El código para **\_change\_pid.c** se muestra en la Fig. 6.



Fig(6):\_change\_pid.c

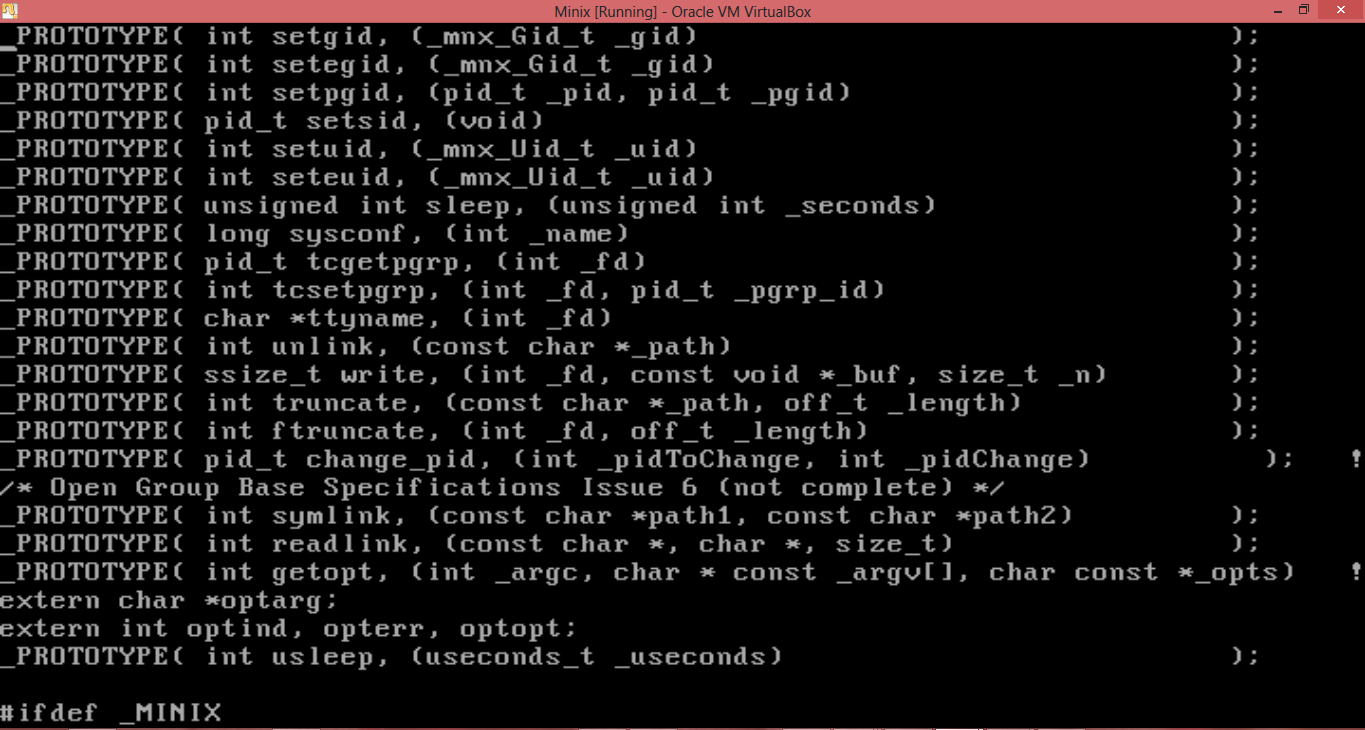
Para compilar la biblioteca agregamos el nombre del archivo **\_change\_pid.c** en Makefile.in que está en el directorio **/usr/src/lib/posix,** (Fig. 7)se ejecuta el comando **make Makefile** que genera un nuevo archivo makefile con las reglas para el nuevo archivo incluido, nos movemos al directorio **/usr/src/** y ejecutamos el comando **make libraries.**



Fig(6):Makefile.in

Una vez que se ha creado el código binario para el servidor y la nueva biblioteca, es necesario mezclar el código binario y crear una nueva imagen del SO Minix 3. Para ello nos situamos en el directorio **/usr/src/tools**, usamos el comando **make hdboot** y **make install** para crear una nueva imagen de boteo en el directorio /boot/image/.

Para utilizar cualquier llamada al sistema normalmente se incluye un fichero, por ejemplo, cómo se está trabajando con la eliminación de directorios se debe incluir unistd.h. Para hacer esto y evitar el warning que suele salir cuando se compila un fichero donde se utilice la nueva llamada al sistema, se debe modificar el fichero localizado en **/usr/src/include/unistd.h** y **/usr/include/unistd.h** y agregar el prototipo de dicha función. El resultado de hacer esta modificación se muestra en la Fig. 7. y a continuación se tiene que crear la nueva imagen del SO, cuyos pasos se han descrito anteriormente.



Fig(7):unistd.h

Ahora veremos un ejemplo de cómo ejecutar la llamada al sistema desde un programa, el código correspondiente se verá en la figura 8



Fig(8):Ejemplo.c

Conclusiones

Hemos desarrollado una nueva llamada al sistema para el Sistema Operativo Minix 3, con la cual nos hemos familiarizado más con dicho sistema, además de profundizar en el lenguaje C.

Bibliografía

* Operating Systems Design and Implementation. Third Edition (formato electrónico).
* Unix Systems Programming (Formato Eléctronico).
* The C Programming Language.