PCSpim's file system access functions

by Oren Weiss

On Windows (NT/2000/XP), Spim will call the ANSI functions _open(...), _read(...), _write(...), _close(...). These functions are declared in io.h and are well documented. The flags that _open uses are defined in fcntl.h. The file permission mode flags are defined in sys/types.h.

If any of this information is incorrect or incomplete, please let me know. At the bottom, I've included a sample program to help you get started. Feel free to tell me if this has helped you.

--SYSCALL 13--

Description:
Opens file

Arguments:
$a0=filename, $a1=flags, $a2=pmode

Return:
$v0=file descriptor, -1 if fail

Filename:
Address of a null terminated string. Default directory is the directory of the assembly source file.

Flags:
Flags are combined by ORing them together.
One of the following flags MUST be used to define access mode:

_O_RDONLY	0x0000	Open file for reading only
_O_WRONLY	0x0001	Open file for writing only
_O_RDWR	0x0002	Open file for reading and writing

The following flags are optional:

_O_APPEND	0x0008	Move file pointer to end of file before every write operation.
_O_RANDOM	0x0010	Specifies that caching is optimized for random access from disk.
_O_SEQUENTIAL	0x0020	Specifies that caching is optimized for sequential access from disk.
_O_TEMPORARY	0x0040	Delete file after last handle closes. (You should probably not use this without _O_CREAT, as this will delete the file on close)
_O_NOINHERIT	0x0080	Prevents creation of a shared file descriptor.
_O_CREAT	0x0100	Create and open file. Has no effect if file exists.
_O_TRUNC	0x0200	Opens a file and truncates it to zero length; must have write permission. _O_TRUNC used with _O_CREAT opens an existing file or creates a new file. THIS DESTROYS THE CONTENTS OF THE FILE!
_O_EXCL	0x0400	MUST BE USED WITH _O_CREAT; Returns an error if file exists
_O_TEXT	0x4000	Opens file in the text (translated) mode. This means that

<CR><LF> will be translated to <LF> in memory on read. Conversely, <LF> will be translated to <CR><LF> on write.

_O_SHORT_LIVED 0x1000 Temporary storage file and, if possible, do not flush to disk.
(I actually have no idea what this does...)
_O_BINARY 0x8000 Opens file with no translation.

Pmode:
Pmode is required only when _O_CREAT is specified. If file already exits, pmode is ignored.
Otherwise, pmode specifies the file permissions. In Windows NT, all files are readable, so write-only permission is not available
[_S_IWRITE is the same as _S_IREAD | _S_IWRITE.]
_S_IREAD 0x0100 Read permission assigned to owner.
_S_IWRITE 0x0080 Write permission assigned to owner.

File Descriptor:
The file descriptor will return in $v0 (NOT $a0!) This integer is used for the other file access functions. A value of -1 indicates the function failed.

--syscall 14--

Description:
Read file

Arguments:
$a0=file descriptor, $a1=buffer, $a2=length (in bytes)

Return:
$v0=bytes read, -1 if fail

File Descriptor:
The file descriptor returned from the open function.

Buffer:
Address of a location in memory where file will be read to.

Length:
The number of bytes to read into memory

Bytes Read:
The number of bytes actually read into memory. -1 if the function failed.

--syscall 15--

Description: Write file

Arguments: $a0=file descriptor, $a1=buffer, $a2=length (in bytes)
Return: $v0=bytes written, -1 if fail

File Descriptor: The file descriptor returned from the open function.

Buffer: Address of a location in memory of data to be written to file.
Length: The number of bytes to write to the file.

Bytes Written: The number of bytes actually written to the file. -1 if the function failed.

--syscall 16--

Description:
Close file

Arguments:
$a0=file descriptor

Return:
0 if successful; -1 if fail

File Descriptor:
The file descriptor returned from the open function.

ALWAYS CLOSE YOUR FILES!!!

For your enjoyment... here is an example program. It will read the first 4 bytes from test.txt and append a copy to the end of the file. If test.txt does not exist, it will create it and store 0000 in the file. The file will be created in the same directory as the assembly source file.

```
.data
buffer:        .word    0x30303030  #ASCII encoded
filename:    .asciiz    "test.txt"

.text
main:

# Open file
la    $a0, filename
li    $a1, 0x010A     # Creates file with read and write... appends
                     # to the end of the file if it exists
li    $a2, 0x0180     # Makes sure we can write to the file if it
has to be created
li    $v0, 13
syscall

# Read 4 bytes into buffer
move  $a0, $v0        # Move the file descriptor to $a0
la    $a1, buffer
li    $a2, 4
li    $v0, 14
syscall

# Write the buffer to the end of the file
la    $a1, buffer
li    $a2, 4
li    $v0, 15
syscall

# Close the file
li    $v0, 16
syscall

# Exit program
li    $v0, 10
syscall
```