Device Implementation in UWIN

Device structure:

The data structure used in UWIN for the device implementation is defined as *devtab.* The definition is this is as shown below.

```
typedef struct devtab
{
   int      minor;
   unsigned char   type;
   void        (*initfn)(struct devtab*);
   char        *name;
   HANDLE (*openfn)(struct devtab*,Pfd_t*, struct Info*, int, HANDLE*);
   ...
} Devtab_t;
```

Minor	Minor number associated with the device.
Type	Type of the device. The device can be a console, pty, tty, etc.
Initfn	function pointer to device initializer. This is an initialization function, for initializing the device.
Name	name of the device
Openfn	function pointer to device opener. This is a function used for opening the device. Each device has a different initializing & opening functions uniquely associated with each. Function pointers are used to point to each of these functions in order to make the function call more generic. Depending upon the initialization function the corresponding devic e is opened using the openfn pointer.

Shared table entries:

UWIN's global memory allocates or maps some shared memory for each of the process initiated by UWIN. This allocation is done through the structure pshare, which shows the entries specific to devices.

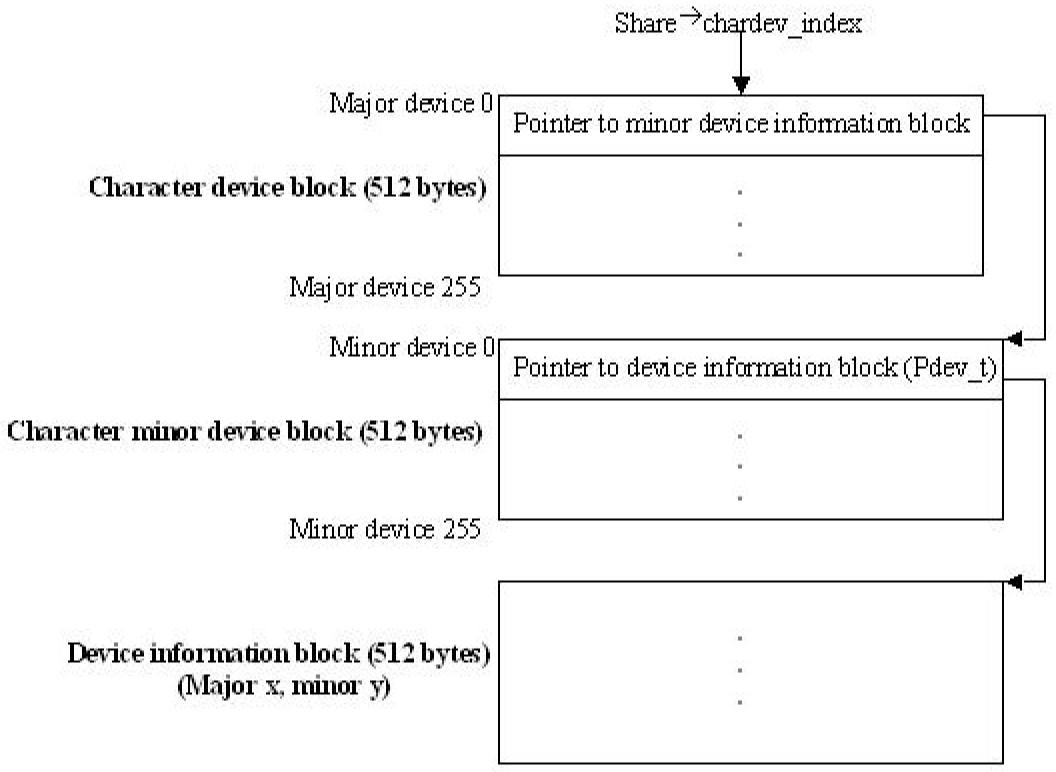
```
typedef struct Pshare
{
...
unsigned short  blockdev_index;
unsigned short  blockdev_size;
unsigned short  chardev_index;
unsigned short  chardev_size;
...
} Shmem_t;
```

blockdev_index	a pointer pointing to the beginning of the block device block.
blockdev_size	gives the number of block devices supported by UWIN.
Chardev_index	a pointer Pointing to the beginning of the character device block.
Chardev_size	gives the number of character devices supported by UWIN.

This is the structure used by UWIN for indexing into a device. Every device associated with UWIN must have an entry linked to this structure.

Architecture:

The figure shows the memory layout allocated by UWIN for a device.


Share→chardev_index
Major device 0
Pointer to minor device information block
Character device block (512 bytes)
Major device 255
Minor device 0
Pointer to device information block (Pdev_t)
Character minor device block (512 bytes)
Minor device 255
Device information block (512 bytes)
(Major x, minor y)

The memory architecture consists of 3 blocks. It is a three-stage indirection used in UWIN. The topmost block is the major device block; the middle one is the minor device block & the last block contains the actual device information. Each block is allocated dynamically from the free block space of global memory. The minor & major blocks will be allocated when UWIN is initialized, and the device information block will be allocated as soon as a device gets initialized. This is the memory where the actual device structure information is stored.

The major & minor device blocks are each of 512 bytes, each storing a 32-bit value. This sets the maximum number of major & minor device to 256. UWIN now supports 13 character devices & 2 block devices.

The major device block will have a pointer to minor device block. This pointer will point to the list of devices found under each device. For example, an entry will be made for tty in the major device block & all the tty's, tty0, tty1 etc will have an entry in the minor device block. Each device will have device structure in the device information block, which points to the reference counts, handles etc that will be allocated when the device is invoked.

The scenario is different when it comes to pty's. In pty's UWIN is dealing with 2 devices; one is on the master side & the other on the slave side. Because of this, UWIN supports only 128 pty's, which is half the number of ttty devices. In the memory architecture also dividing the blocks into half reflects this change. The upper half stores the master device information & the lower half stores the slave information. So for every entry in a major device block, it will have one pointer to master & another pointing to the slave in the minor device block. Each of these master and slave devices will have a different set of memory allocated in the device information block. Telnet is an example where in pseudo terminals find its implementation.

Character devices:

List of character devices supported by UWIN, 13 in number.

Character devices:

0 null
1 tty
2 console
3 pty
4 window
5 clipboard
6 ttyclone
7 ptyclone
8 floppy
9 modem
10 fd
11 printer
12 zero

(The numbers associated with the devices gives the major number of each respectively).

The structure defining these devices is as follows:

```
Devtab_t Chardev[] =
{
    { 1, FDTYPE_NULL, 0, NULL, "" },
    { 256, FDTYPE_TTY, 1, init_tty, "" },
    { 256, FDTYPE_CONSOLE, 1, init_console, "" },
    { 128, FDTYPE_PTY, 1, init_pty,"" },
    { 1, FDTYPE_WINDOW, 0, NULL, "" },
    { 1, FDTYPE_CLIPBOARD,0, init_clipboard, "" },
    { 1, FDTYPE_TTY, 0, init_devtty, "" },
    { 1, FDTYPE_PTY, 0, init_devpty, "" },
    { 2, FDTYPE_FLOPPY,0, NULL, "\\\\.\\%c:\0" },
    { 8, FDTYPE_MODEM,1,    init_modem, "com%d\0" },
    { OPEN_MAX, FDTYPE_DEVFD, 0, init_devfd, "" },
    { 8, FDTYPE_LP, 0, init_lp, "lpt%d:" },
    { 1, FDTYPE_ZERO, 0, NULL, "" },
};
```

The first entry gives the number of devices supported, the second fields is the device type which are the #define equivalents; the fourth field gives the function pointer for each device, the last field gives the name of the device, the way windows recognize the device. The order in the structure is very important as it corresponds to the major number of the device, if a new device has to be inserted then it must appear in the bottom of the structure, after all the existing entries.

Block devices:

Block devices supported by UWIN,

Block devices:
 0 disk
 1 tape

The structure defining these devices is as follows:

```
Devtab_t Blockdev[] =
{
    {1, FDTYPE_FILE, 0, NULL, "\\\\.\\PhysicalDrive%d"},
    {128, FDTYPE_TAPE, 1, init_tape, ""},
};
```

As in character devices, here also the number associated with the device gives the major number of the device.

Functions & macros used:

devtab_ptr()

Returns a pointer to the specified minor device table given the major number and the device index (e.g. chardev_index)

Given the major number & whether it's a character device or block device, this function returns a pointer to that minor device block where more information about a device can be stored.

dev_ptr()

Returns pointer to the device structure (Pdev_t) given the device slot number

These functions & macros are useful when it is required to find the reference count of a device or if it is required to write to a device through the handle.

Thread association:

Each of the character devices is associated with threads. These devices involve the input/output operation for UWIN.

- Consoles (Console Window Application)
    - –/dev/ttyXY (256 nos.)
    - –ReadConsoleThread()
    - –WriteConsoleThread()

Consoles run from tty10 to tty 266 (only 256 are allowed). Even the mouse movements are also recorded for editing using cut 7 paste options.

- TTYs (Dumb Terminals)
    - –/dev/ttyXY (10 nos.)
    - –ReadTtyThread()
    - –WriteTtyThread()

tty run from tty0 to tty9. The data structure allows 256 devices but it is being restricted only to 10 devices. This depends on the number of com ports available & there are only 2 ports available and in this scenario 10 is sufficient. Because of this only two variables XY are allocated. The two threads are used to read & write into the com ports.

- PTYs (Pseudo-terminals)
    - –/dev/ptyXY and /dev/ttyXY (128 nos. each)
    - –ReadPtyThread()
    - –WritePtyThread()

The naming convention used in pty follows hexa decimal convention. It starts from ptyp0 to ptypf and again in starts from ptyq0 to ptyqf.