The *ioctl* System Call

- The ***ioctl*** function has traditionally been the system interface used for any operation that required controlling and conducting device input/output.

- Every device can have its own ***ioctl*** commands, which can be read ***ioctl***s (to send information from a process to the kernel), write ***ioctl***s (to return information to a process), both, or neither.

- The ***ioctl*** function is called with three parameters: the file descriptor of the appropriate device file, the ***ioctl*** number, and a parameter, which is of type long so you can use a cast to use it to pass anything.

- Posix is getting rid of ***ioctl***, by creating specific wrapper functions to replace ***ioctl***s whose functionality is being standardized by Posix.

- For example, the Unix terminal interface was traditionally accessed using ***ioctl*** but Posix.1 created 12 new functions for terminals: ***tcgetattr*** to get the terminal attributes, ***tcflush*** to flush pending input or output, and so on.

- Posix.lg is replacing one ***ioctl***: the new ***sockatmark*** function replaces the **SIOCATMARK** ***ioctl***.

- Several ***ioctl***s remain for implementation-dependent features related to network programming: obtaining the interface information, and accessing the routing table and the ARP cache, for example.

- We will discuss the ***ioctl*** requests related to network programming. A common use of ***ioctl*** by network programs is to obtain information on all the host's interfaces when the program starts: the interface addresses, whether the interface supports broadcasting, whether the interface supports multicasting, and so on.

- This function manipulates the device parameters using a file descriptor:

  #include <sys/ioctl.h>

  int ioctl (int fd, int request, void *argp);

- ***fd*** is an open file descriptor.

- The argument ***request*** describes a variety of operations depending on the category of the ***ioctl***.

- The third argument is always a pointer, but the type of pointer depends on the *request*.

- We can divide the ***requests*** related to networking into six categories.

  - socket and interface operations
  - file operations
  - ARP cache operations
  - routing table operations
  - streams system

- The table below lists all the values ***request,*** can take, along with the **datatype** of what the ***argp*** address must point to.

- Note that there are many more socket and interface type requests defined in **ioctl.h**. I have just listed the more commonly used ones.

Category	***request***	**Description**	**Datatype**
socket & interface	SIOCGIFCONF	get list of all interfaces	struct ifconf
	SIOCSIFADDR	set interface address	struct ifreq
	SIOCCIFADDR	get interface address	struct ifreq
	SIOCSIFFLACS	set interface flags	struct ifreq
	SIOCGIFFLAGS	get interface flags	struct ifreq
	SIOCSIFDSTADDR	set point-to-point address	struct ifreq
	SIOCGIFDSTADDR	get point-to-point address	struct ifreq
	SIOCGIFBRDADDR	get broadcast address	struct ifreq
	SIOCSIFBRDADDR	set broadcast address	struct ifreq
	SIOCGIFNETMASK	get subnet mask	struct ifreq
	SIOCSIFNETMASK	set subnet mask	struct ifreq
	SIOCCIFMETRIC	get interface metric	struct ifreq
	SIOCSIFMETRTC	set interface metric	struct ifreq
	SIOCSIFHWADDR	set MAC address	struct ifreq
	SIOCGIFHWADDR	get MAC address	struct ifreq
file	FIONBIO	set/clear nonblocking flag	int
	FIOASYNC	set/clear asynchronous I/O flag	int
	FIONREAD	get # bytes in receive buffer	int
	FIOSETOWN	set PID or process GID of file	int
	FIOGETOWN	get PID or process GID of file	int
ARP	SIOCSARP	set ARP entry	struct arpreq
	SIOCGARP	get ARP entry	struct arpreq
	SIOCDARP	delete ARP entry	struct arpreq
routing	SIOCADDRT	add route	struct rtentry
	SIOCDELRT	delete route	struct rtentry
streams	(see text)		

File Operations

- This group of requests begins with FIO and may apply to certain types of files, in addition to sockets.

- The following are only the requests that apply to sockets. All of the requests require that the third argument to ***ioctl*** point to an integer.

- **FIONBIO**

    - The nonblocking flag for the socket is cleared or turned on, depending whether the third argument to ***ioctl*** points to a zero or nonzero value, respectively

- **FIOASYNC**

    - The flag that governs the receipt of asynchronous I/0 signals (**SIGIO**) for the socket is cleared or turned on, depending whether the third argument to ***ioctl*** points to a zero or nonzero value, respectively

- **FIONREAD**

    - Return in the integer pointed to by the third argument to ***ioctl*** the number of bytes currently in the socket receive buffer.

    - This feature also works for files, pipes, and terminals.

- **FIOGETOWN**

    - Return through the integer pointed to by the third argument either the PID or the process GID that is set to receive the SIGIO or SIGURG signal for this socket.

- **FIOSETOWN**

    - Set either the PID or the process GID to receive the SIGIO or SIGURG signal for this socket from the integer pointed to by the third argument.

Socket and Interface Operations

- The first step employed by many programs that deal with the network interfaces on a system is to obtain from the kernel all the interfaces configured on the system.
- This is done with the **SIOCGIFCONF** request, which uses the ***ifconf*** structure, which in turn uses the ***ifreq*** structure, both of which are described in **net/if.h**.
- Before calling ***ioctl*** we allocate a buffer and an ***ifconf*** structure and then initialize the latter.
- This is illustrated in the diagram in your textbook. Assuming that the buffer size is 1024 bytes.
- The third argument to ***ioctl*** is a pointer to the ***ifconf*** structure. If we assume that the kernel returns two ***ifreq*** structures, we could have the arrangement shown when the ***ioctl*** returns.
- The shaded regions have been modified by ***ioctl***. The buffer has been filled in with the two structures and the ***ifc_len*** member of the ***ifconf*** structure has been updated to reflect the amount of information stored in the buffer.

- The **SIOCGIFCONF** request returns the name and a socket address structure for each interface that is configured. There are a multitude of other requests that we can then issue to set or get all the other characteristics of the interface.
- The ***get*** version of these requests *(***SIOCGXXX***)* is issued by the ***netstat*** program, and the ***set*** version (**SIOCSxxx***)* is issued by the ***ifconfig*** program.
- Any user can get the interface information, while it takes superuser privileges to set the information.
- These requests take or return an ***ifreq*** structure whose address is specified as the third argument to ***ioctl***.
- The interface is always identified by its name: **le0, lo0, eth0, eth1**, or whatever in the ***ifr_name*** member.
- Many of these requests use a socket address structure to specify or return an IP address or address mask with the application.
- For lPv4, the address or mask is contained in the ***sin_addr*** member of an Internet socket address structure (***sockaddr_in***).

- **SIOCGIFADDR**
  - Return the unicast address in the ***ifr_addr*** member.

- **SIOCSIFADDR**
  - Sets the interface address from the ***ifr_addr*** member. The initialization function for the interface is also called.

- **SIOCGIFFLAGS**
  - Return the interface flags in the ***ifr_flags*** member.
  - The names of the various flags are ***IFF_xxx*** and are defined by including the **<net/if.h>** header.
  - The flags indicate, for example, if the interface is up (IFF_UP), if the interface is a point-to-point interface (IFF_POINTOPOINT), if the interface supports broadcasting (IFF_BROADCAST), and so on.

- **SIOCSIFFLAGS**
  - Set the interface flags from the ***ifr_flags*** member.

- **SIOCGIFDSTADDR**
  - Return the point-to-point address in the ***ifr_dstaddr*** member.
- **SIOCSIFDSTADDR**
  - Set the point-to-point address from the ***ifr_dstaddr*** member.

- **SIOCGIFBRDADDR**
  - Return the broadcast address in the ***ifr_broadaddr*** member.
  - The application must first fetch the interface flags and then issue the correct request: **SIOCGIFBRDADDR** for a broadcast interface or **SIOCGIFDSTADDR** for a point-to-point interface.

- **SIOCSIFBRDADDR**
  - Set the broadcast address from the ***ifr_broadaddr*** member.

- **SIOCGIFNETMASK**
  - Return the subnet mask in the ***ifr_addr*** member.

- **SIOCSIFNETMASK**
  - Set the subnet mask from the ***ifr_addr*** member.

- **SIOCGIFMETRIC**
  - Return the interface metric in the ***ifr_metric*** member.
  - The interface metric is maintained by the kernel for each interface but is used by the routing daemon routed. The interface metric is added to the hop count (to make an interface less favorable).

- **SIOCSIFMETRIC**
  - Set the interface routing metric from the ***ifr_metric*** member.

- **SIOCGIFHWADDR**
  - Return the MAC address in the ***ifr_hwadd*** member.

- **SIOCSIFHWADDR**
  - Set the MAC address through the ***ifr_hwadd*** member.

ARP Cache Operations

- The ARP cache is also manipulated with the ***ioctl*** function.
- These requests use an ***arpreq*** structure, shown below and defined by including the <**net/if_arp.h**> header.

```
struct arpreq
{
        struct  sockaddr arp_pa;        /* protocol address */
        struct  sockaddr arp_ha;        /* hardware address */
        int     arp_flags;              /* flags */
};

#define         ATF_INUSE       0x01    /* entry in use */
#define         ATF_COM         0x02    /* completed entry (hardware addr valid) */
#define         ATF_PERM        0x04    /* permanent entry */
#define         ATF_PUBL        0x08    /* published entry (respond for other host)*/
```

- The third argument to ***ioctl*** must point to one of these structures. The following three ***requests*** are supported:

- **SIOCSARP**
  - Add a new entry to the ARP cache or modify an existing entry.
  - ***arp_pa*** is an Internet socket address structure containing the IP address and
  - ***arp_ha*** is a generic socket address structure with ***sa_family*** set to **AF_UNSPEC** and ***sa_data*** containing the hardware address (e.g., the 6-byte Ethernet address).
  - The two flags **ATF_PERM** and **ATF_PUBL** can be specified by the application. The other two flags, **ATF_INUSE** and **ATF_COM**, are set by the kernel.

- **SIOCDARP**
  - Delete an entry from the ARP cache. The caller specifies the Internet address for the entry to be deleted.

- **SIOCGARP**
  - Get an entry from the ARP cache. The caller specifies the Internet address and the corresponding Ethernet address is returned along with the flags.
  - Only the superuser can add or delete an entry. These three requests are normally issued by the ***arp*** program.
  - These ARP-related ***ioctl*** requests are not supported on some newer systems, which use routing sockets for these ARP operations.

- Notice that there is no way with ***ioctl*** to list all the entries in the ARP cache. Most versions of the ***arp*** command, when invoked with the -a flag (list all entries in the ARP cache), read the kernel's memory ***(/dev/kmem***) to obtain the current contents of the ARP cache.

Routing Table Operations

- Two ***ioctl*** requests are provided to manipulate the routing table. The two requests require that the third argument to ***ioctl*** be a pointer to an ***rtentry*** structure defined in the <**net/route.h**>.

- **SIOCADDRT**
  - Add an entry to the routing table.

- **SIOCDELRT**
  - Add an entry to the routing table.

  - There is no way with ***ioctl*** to list all the entries in the routing table.
  - The program obtains the routing table by reading the kernel's memory **(/dev/kmem).**

- The examples given show how we can use ***ioctl*** to get detailed information on all the network interfaces on a system.