Elementary Name and Address Conversions

Elementary Names & Address Conversions – 1/3

- **DNS (Domain Name System) used primarily to map between hostnames and IP addresses**
- Hostname can be simple name, *google*, or a fully qualified domain name (FQDN) such as *www.google.com*
- **/etc/hosts** – file which consists all the host details

Elementary Names & Address Conversions – 2/3

- Resource records
  - **A :- An A record maps a hostname into a 32-bit IPv4 address**
  - AAAA :- Also called a "quad A" record maps a hostname into a 128-bit IPv6 address
  - PTR :- Called "pointer records" maps IP addresses into hostnames.
  - **CNAME :- "Canonical Name"The official name of the host**

Elementary Names & Address Conversions – 3/3

- Functions
  - gethostbyname()
  - getservbyname()
  - getservbyport()
  - gethostname()
  - gethostbyaddr()
  - uname()

gethostbyname() – 1/5

- **Looks up a hostname, given its name**

```
#include <netdb.h>
struct hostent  *gethostbyname (const char
*hostname);
```

- Returns
  - Non-null pointer if successful
  - NULL on error with h_errno set

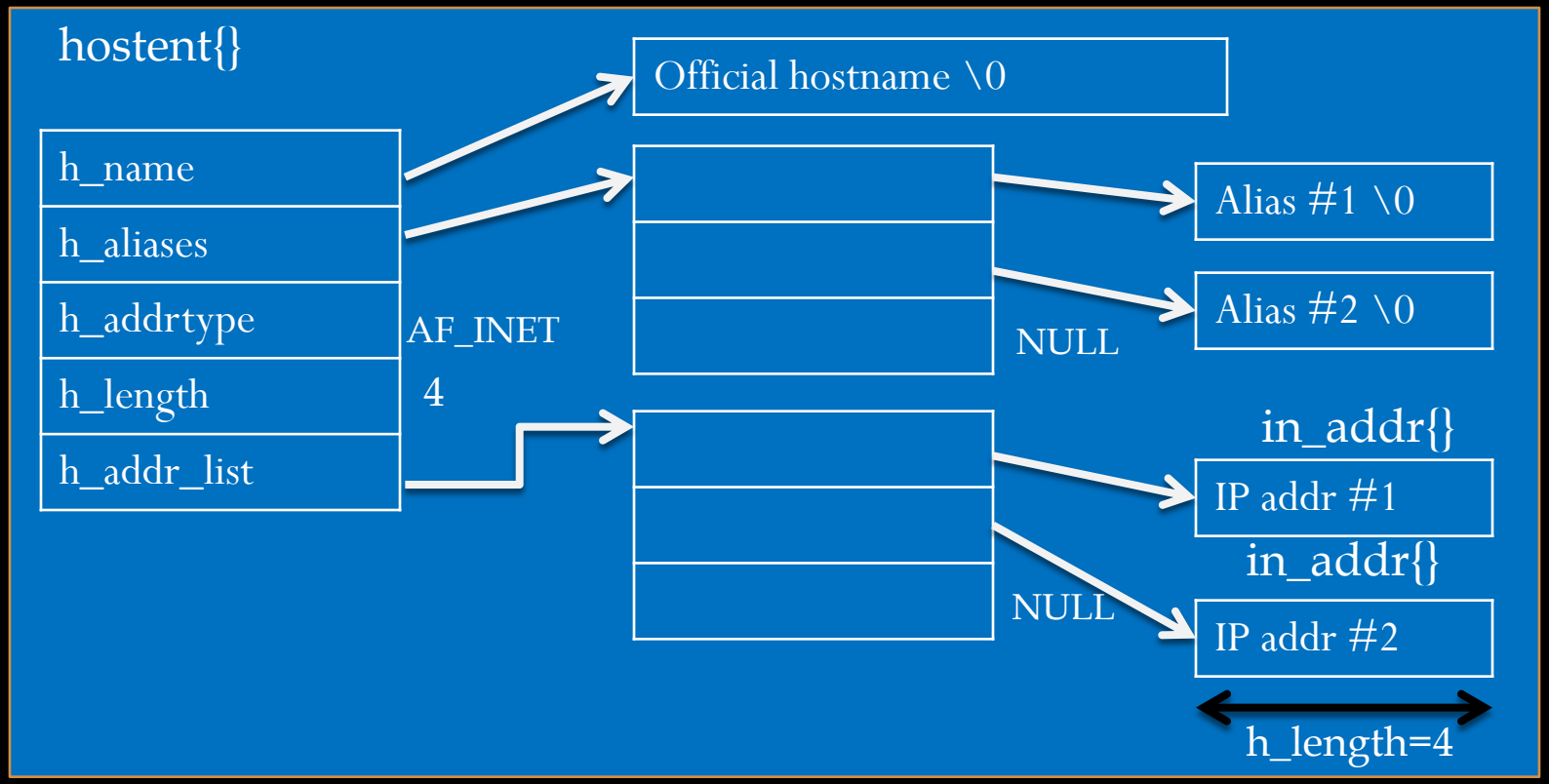
gethostbyname() – 2/5

- The non-null pointer returned is a **hostent** structure with following details

```
struct hostent {
    char *h_name;  //official (canonical) name of host
    char **h_aliases;  //pointer to array of pointers to alias names
    int h_addrtype;  //host address type : AF_INET
    int h_length;  //length of address
    char **h_addr_list; //ptr to array of ptrs with IP addresses
    };
```

gethostbyname() – 3/5

- hostent structure and the information it contains



gethostbyname() – 4/5

- If error is there, the function sets the global integer h_errno to one of the following constants defined by <netdb.h>
  - HOST_NOT_FOUND
  - NO_DATA (name is valid, but does not have A record)
  - TRY_AGAIN
  - NO_RECOVERY

gethostbyname() – 5/5

- SAMPLE PROGRAM

getservbyname() – 1/6

- Services, like hosts, are often known by names too
- If we refer to a service by its name in our code, instead of its port number, and if mapping from name to port number is contained in **/etc/services** file, then if port number changes, we need to modify only one line in the file
- This function **looks up a service given its name**

getservbyname() – 2/6

```
#include <netdb.h>
struct servent * getservbyname (const char *servname,
const char *protoname);
```

- Returns
  - Non null pointer if OK
  - NULL on error

getservbyname() – 3/6

- The servent structure is defined as follows

```
struct servent {
    char *s_name; //official service name
    char **s_aliases; //alias list
    int s_port; //port no
    char *s_proto; //protocol to use
    };
```

getservbyname() – 4/6

- **Service name *servname* must be specified**
- If protocol is also specified, then entry must also have matching protocol
- **If *protocol* is not specified, and service supports multiple protocols, it is dependent on the *port no* returned**
- **Main field of interest is *port no***

getservbyname() – 5/6

- Typical calls to this function

```
struct servent *sptr;
sptr=getservbyname("domain","udp"); //DNS using UDP
sptr=getservbyname("ftp","tcp"); //FTP using TCP
sptr=getservbyname("ftp",NULL); //FTP using TCP
sptr=getservbyname("ftp","udp"); //call will fail
```

The fourth call fails because FTP supports only TCP

getservbyname() – 6/6

- Try to write the program by yourself…
- Refer to gethostbyname() program

getservbyport() – 1/2

- **Looks up a service given its port number and optional protocol**

```
#include <netdb.h>
struct servent *getservbyport (int port, const char
*protoname);
```

- Returns
  - Non null pointer if OK
  - NULL on error

getservbyport() – 2/2

- Calls to this function

```
struct servent *sp;
sp=getservbyport(htons(53), “udp”); //DNS using UDP
sp=getservbyport(htons(21), “tcp”); //FTP using TCP
sp=getservbyport(htons(21), NULL); //FTP using TCP
sp=getservbyport(htons(21), “udp”); //call will fail
```

Call fail because there is no service that uses port no 21 with UDP

gethostbyaddr() – 1/2

- Takes a binary address and tries to **find the hostname corresponding to that binary address**

- Reverse of gethostbyname

```
#include <netdb.h>
struct hostent *gethostbyaddr(const char *addr, size_t len, int family);
```

- Return value
  - Pointer to hostent structure if OK
  - h_errno if error

gethostbyaddr() – 2/2

- The *addr* is not a char* but it is a pointer to in_addr or in6_addr structure containing IPv4 or IPv6 address
- The *len* is size of this structure. 4 for IPv4 and 16 for IPv6 addresses
- The *family* is either AF_INET or AF_INET6

Kindly go through the next slides

gethostname() – 1/2

- **Returns name of current host**

```
#include <unistd.h>
int gethostname(char *name, size_t namelen);
```

- Return value
  - 0 if OK
  - -1 on error

gethostname() – 2/2

- *name* is pointer to where hostname is stored
- *namelen* is size of array
- Maximum size of hostname is normally MAXHOSTNAMELEN constant defined by including the <sys/param.h> header

uname()

- **Returns name of current host**

  #include <sys/utsname.h>

  int uname (struct utsname **name*);

- This function fills in a utsname structure whose address is passed by caller