Chapter 14

Application Layer

- Client-Server Model
- Concurrency
- Processes

Figure 14-1

Comparison between OSI and TCP/IP

- Application
- Presentation
- Session
- Transport
- Network
- Data link
- Physical

OSI model

TCP and UDP
IP
Data link
Physical

TCP/IP protocol suite
Slides from *TCP/IP* - Forouzan

**Figure 14-2**  
Client-server model

**Figure 14-3**  
Client-server relationship
Figure 14-4

Server types

- Connectionless iterative
- Connectionless concurrent
- Connection-oriented iterative
- Connection-oriented concurrent

Figure 14-5

Connectionless iterative server

Client
Ephemeral port
UDP
Client
Ephemeral port
UDP

One client at a time

Server
Well-known port
UDP
Client
Ephemeral port
UDP

Client
Ephemeral port
UDP
Figure 14-6: Connection-oriented concurrent server

Child server

Client

Ephemeral port

TCP

Parent server

Child server

Child server

Ephemeral port

Well-known port is used only for connection establishment

Figure 14-7: Classes and objects

```
class {
    private:
    int x;
    int y;
    public:
    ........;
    ........;
};
```

Object 3:

```
4 6
x y
```

Object 2:

```
10 -3
x y
```

Object 1:

```
-3 24
x y
```
int main (void) {
    /* Declaration */
    int x ;
    int y ;
    /* Statements */
    .....................
    .....................
    .....................
    .....................
    .....................
    .....................
    return 0;
}

Figure 14-8
Program and processes

Program

Figure 14-9
Prototype for the getpid function

pid_t getpid (void) ;
A program that prints its own process id

```c
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>

int main (void)
{
    printf ("My process id is %ld", (long) getpid());
    return 0;
}
```

Prototype for the fork function

```c
pid_t fork (void);
```
```c
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>

int main (void)
{
    printf ("Hello World\n") ;
    fork () ;
    printf ("Bye World\n") ;
    return 0;
}
```

Figure 14-12  A program with one parent and one child

```
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>

int main (void)
{
    printf ("Parent\n") ;
    fork () ;
    printf ("Parent and first child\n") ;
    fork () ;
    printf ("Parent, first child, second child and grandchild");
    return 0;
}
```

Figure 14-13  A program with two fork functions
### Figure 14-14
The output of the program in Figure 14-13

<table>
<thead>
<tr>
<th>Parent</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parent and first child</td>
</tr>
<tr>
<td>Parent, first child, second child, and grandchild</td>
</tr>
</tbody>
</table>

### Figure 14-15
A program that prints the process ids of the parent and the child

```c
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>

int main (void)
{
    pid_t pid;
    pid = fork();
    if (pid != 0 )
        printf ("Parent process id is %ld", (long) getpid());
    else
        printf ("Child process id is %ld", (long) getpid());
    return 0;
}
```
void main (void)
{
    .........
    .........
    pid_t pid;
    for ( ; )
    {
        Connection from client
        pid = fork ();
        if (pid != 0)
        {
            Code for parent
        }
        else
        {
            Code for child
        }
    }
}

Example of a server program with parent and child processes

Figure 14-16

Forouzan Notes 14-17
COSC 6377 - Fall 2000