3.1 What are the benefits and the disadvantages of each of the following? Consider both the system level and the programmer level.

a. Synchronous and asynchronous communication

ANS:

A benefit of synchronous communication is that it allows a rendezvous between the sender and receiver. A disadvantage of a blocking send is that a rendezvous may not be required and the message could be delivered asynchronously. As a result, message-passing systems often provide both forms of synchronization.

答案可參考第八版課本 P.122 ，第七版 P.99

b. Automatic and explicit buffering

ANS:

Automatic buffering provides a queue with indefinite length, thus ensuring the sender will never have to block while waiting to copy a message. There are no specifications on how automatic buffering will be provided; one scheme may reserve sufficiently large memory where much of the memory is wasted. Explicit buffering specifies how large the buffer is. In this situation, the sender may be blocked while waiting for available space in the queue. However, it is less likely that memory will be wasted with explicit buffering.

答案可參考第八版課本 P.122~123 , 第七版 P.99

c. Send by copy and send by reference

ANS:

Send by copy does not allow the receiver to alter the state of the parameter; send by reference does allow it. A benefit of send by reference is that it allows the programmer to write a distributed version of a centralized application. Java's RMI provides both; however, passing a parameter by reference requires declaring the parameter as a remote object as well

d. Fixed-sized and variable-sized messages

ANS:

The implications of this are mostly related to buffering issues; with fixed-size messages, a buffer with a specific size can hold a known number of messages. The number of variable-sized messages that can be held by such a buffer is unknown. Consider how Windows 2000 handles this situation: with fixed-sized messages (anything < 256 bytes), the messages are copied from the address space of the sender to the address space of the receiving process. Larger messages (i.e. variable-sized messages) use shared memory to pass the message.

3.11 Using the program in Figure 3.29, identify the values of pid at lines A, B, C and D. (Assume that the actual pids of the parent and child are 2600 and 2603, respectively.)

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

int main ()
{
    /* fork a child process */
    pid = fork();
    if ( pid < 0 ) { /*error occurred
        fprintf(stderr, "Fork Failed");
        return 1;
    }
    else if (pid ==0) { /* child process */
        pid1 = getpid();
        printf("child: pid = %d" ,pid); /* A */
        printf("child: pid1 = %d" ,pid1); /* B */
    }
    else { /* parent process */
        pid1 = getpid();
        printf("parent: pid = %d" ,pid); /* C */
        printf("parent: pid1 = %d" ,pid1); /* D */
        wait(NULL);
    }
    return 0;
}
```

Figure3.29 what are the pid values?

ANS:

A = 0, B = 2603, C = 2603, D = 2600

3.13 The Fibonacci sequence is the series of numbers 0, 1, 1, 2, 3, 5, 8 …. Formally, it can be expressed as:

$fib_0 = 0$

$fib_1 = 1$

$fib_n = fib_{n-1} + fib_{n-2}$

Write a C program using the fork() system call that generates the Fibonacci sequence in the child process. The number of the sequence will be provided in the command line. For example if 5 is provided, the first five numbers in the Fibonacci sequence will be output by the child process. Because the parent and child processes have their own copies of the data, it will be necessary for the child to output the sequence. Have the parent invoke the wait() call to wait for the child process to complete before exiting the program. Perform necessary error checking to ensure that a non-negative number is passed on the command line.

題目重點在於 fork 和 wait

答案可參考第八版課本 P.113，第七版 P.89