Block I/O

fread and *fwrite* functions are the most efficient way to read or write large amounts of data.

fread() – reads a specified number of bytes from a binary file and places them into memory at the specified location.

prototype for fread ( ):

```
int fread(void *InArea, int elementSize, int count, FILE *fp);
```

InArea – a *pointer to the input area in memory*.
elementSize – *size of a basic data element*, often specified using the sizeof operator
count – *number of elements*
fp – *file pointer of an open file*.

fread returns the number of items read.

Block I/O

fwrite – writes the specified bytes of data from memory to the output file.

prototype for fwrite ( ):

```
int fwrite(void *OutArea, int elementSize, int count, FILE *fp);
```

OutArea – *a pointer to memory* holding the data to be written.

elementSize – *size of a basic data element*, often specified using the sizeof operator

count – *number of data elements*

fp – *file pointer of an open file*.

fwrite copies elementSize * count bytes from the address specified by OutArea to the file.

fwrite returns the total number of characters written.

Block input and output

Here is a more efficient implementation of the *cat-like* program that copies standard input to the standard output.

```
/* p12.c */
#include <stdio.h>
main()
{
    unsigned char *buff;
    int len = 0;
    int iter = 0;
    buff = malloc(1024);
    if (buff == 0)
    exit(1);
    while ((len = fread(buff, 1, 1024, stdin)) != 0)
    {
       fwrite(buff, 1, len, stdout);
       fprintf(stderr, "%d %d \n", iter, len);
       iter += 1;
    }
}
0 322
```

Block input and output

Questions: Removing the parentheses surrounding

```
(len = fread(buff, 1, 1024, stdin))
```

will break the program. Explain exactly *how and why* things will go wrong in this case?

What will happen if *len* in the *fwrite()* is replaced by 1024?