Bitwise operators

The main bitwise operators are summarised in the table below.

Condition	Meaning	Condition	Meaning
a & b	a 'and' b	a >> n	right shift a by n
a | b	a 'or' b	a << n	left shift a by n
a ^ b	a 'exclusive or' b		

These operators are typically used with integer variable types or signal bytes stored in char variables. They act on the binary form of the number and are typically used for bit packing or testing packed bits. For example, if the status of several switches needs to be read, their input could be stored in one integer variable.

As revision of the second tutorial, the decimal values of each bit can be printed with the program below:

```
#include <stdio.h>
int main() {
  int bit = 0, i = 1;
  while(i>0) { /* Loop until the sign bit is set */
    printf(" pow(2,%2d) = %11d\n",bit,i);
    i = i<<1; /* Shift value of i left by one. */
    bit++; /* Increment the counter by one. */
  }
  return 0; /* Return success to the operating system. */
}
```

In this example, the value stored in the variable i is shifted one place to the left. The left shift operator has the effect of moving all of the bits in the variable i one place to the left. If a bit is shifted outside the memory allocation of the variable i, the bit is lost. In this case, i only contains one. Therefore, the action of the left shift operator is to move to the next power of two. When the bit in the variable i is moved into the sign bit the number becomes negative which causes the while loop to stop.

The & operator is very useful for testing if bits are set. This can be combined with the left or right shift operator to test every bit in a integer variable,

```
#include <stdio.h>
int main() {
  char str[33]; /* Declare a character array to hold the output. */
  int bit, i = 235643; /* Declare a number to convert to binary. */
  for(bit=31;bit>0;bit--) { /* Loop from left to right */
    if(((1<<bit) & i) == 0) str[31-bit] = '0'; /* False */
    else str[31-bit] = '1'; /* True */
  }
  str[32]='\0'; /* Add the string terminator */
  printf("%d (decimal) = %s (binary)\n", i, str);
  return 0; /* Return success to the operating system. */
}
```

In this example program, each character in the char array is set according to the binary value. Then to complete the string, the string terminator is added. Finally, the binary form of the integer number is printed.

System commands

It can be useful to be able to run shell commands or other programs without directly linking to an associated library. This can be accomplished with the system function,

```
#include <stdlib.h>
int main() {
  system("ls ./"); /* List the files in current directory. */
  return 0; /* Return success to the operating system. */
}
```