```
        CC          = gcc
        CFLAGS      = -g -ansi -Wall

        # files associated with the bitstring package
        HDRS  = bits.h
        SRCS  = bits.c
        OBJECTS     = bits.o

        # files associated with the test driver
        DHDRS       =
        DSRCS       = driver.c
        DOBJECTS    = driver.o

        # Here is the default target
0       all :: driver

        # This next rule identifies how to make objects from SRCS
1       %.o :  %.c
                $(CC) $(CFLAGS) -c $<

        # this rule generates our test driver
        driver : $(OBJECTS) $(DOBJECTS)
                $(CC) $(CFLAGS) $(OBJECTS) $(DOBJECTS) -o driver

        # The following rule removes unwanted object modules
        clean :
                usr/bin/rm $(OBJECTS) $(DOBJECTS)

        # this rule wraps up stuff in a shar file, for emailing or moving
        shar :
                shar Makefile $(SRCS) $(HDRS) $(DSRCS) $(DHDRS) > bits.shar

        # Here we can wrap stuff up in a tar file
        tar  :
                tar cvf bits.tar Makefile $(SRCS) $(HDRS) $(DSRCS) $(DHDRS)
        #or perhaps you prefer
        #       tar cvf - Makefile $(SRCS) $(HDRS) $(DSRCS) $(DHDRS) > bits.tar

        # The last rule determines the dependencies between code and header files
        # and saves us from having to do it.
        # You should do a make depend after every re-organization of the files
        # that would alter dependencies.
2       depend :
                makedepend $(SRCS) $(DSRCS)
        # DO NOT DELETE THIS LINE -- make depend depends on it.

3       bits.o driver.o :       bits.h    /usr/include/stdio.h
```

```
4               Mention
        make -n all    to see what will be executed

        %.o:            identifying source files and default compile command
        clean: remove object files and executable modules.
        shar:           package source for shipping
        tar:            package source as a directory for shipping.
        -------------------comments on make -n command------------------
        make -n all
                gcc -g -ansi -Wall -c bits.c
                gcc -g -ansi -Wall -c driver.c
                gcc -g -ansi -Wall bits.o driver.o -o driver

        touch driver.c
        make -n all
                gcc -g -ansi -Wall -c driver.c
                gcc -g -ansi -Wall bits.o driver.o -o driver

        touch bits.h
        make -n all
                gcc -g -ansi -Wall -c bits.c
                gcc -g -ansi -Wall -c driver.c
                gcc -g -ansi -Wall bits.o driver.o -o driver
        /*----------------------bits.h----------------------------------*/
        / *
        *       The bitstring header file - general information
        *
        *       Note: the specifications for the procedures and macros are placed
        *       in this header, because this interface is what a user of the
        *       bitstring module sees.  They may not even have access to the
        *       implementation.
        *
        *       General Assumptions:
        *       The bits of a bitstring x are numbered in string order, that is
        *               0, 1, 2, ..., bits_length(x)-1
        *
        *       The length of a bitstring may vary.
        *
        *       An individual bit is an unsigned int of value 0 or 1.  If something
        *       other than 0 or 1 is supplied by the user as a bit, only the least
        *       significant bit is used.
        * /
        / *
        *       A bitstring is of type bits.
        *
        *       Invariant: at present we only permit fixed length bitstrings of
        *               as many bits as are in an unsigned int.
        * /
```

```
5       typedef unsigned int bits;

6       #define bits_length(x) ( (int) (8*sizeof(bits)))
        / *
        *       Get the length of a bitstring
        *               Pre:   none
        *               Post:  bits_length(x) is the number of bits in the bitstring x
        *               Side   Effects: none
        * /
7       extern unsigned int bits_get(bits x, int i);
        / *
        *       Pre:                0 <= i < bits_length(x)
        *       Post:               bits_get(x,i) is the value of bit i of x
        *       Side effects:       none
        * /

        extern bits bits_put(bits x, int i, unsigned int v);
        / *
        *       Pre:                0 <= i < bits_length(x)
        *       Post:               bits_put(x, i, v) is the same as x except that bit i is set
        *                           to the same value as bit 0 of v.
        *       Side effects:       none
        * /
        extern void bits_printf(bits x);
        / *
        *       Pre:                none
        *       Post:               none
        *       Side Effects:       the bits of x are printed on stdout in string order,
        *                                   0, 1, 2, ...
        * /
8       #define bits_from_int(x) (x)
        / *
        *       Convert an int into a bitstring
        *               Pre:    none
        *               Post:   bits_from_int(x) is a bitstring that contains the bits of x
        *                       ordered from least significant first to most significant last.
        *               Side Effects: none
        * /
        /*-----------------------bits.c---------------------------------*/
        / *
        *               The bitstring implementation
        * /
        #include <stdio.h>
        #include "bits.h"

        unsigned int bits_get (bits x, int i) {
                return ( x >> i) & 1;
        }

        bits bits_put (bits x, int i, unsigned int v) {
                if ( v & 1 ) {
                        return x | (1 << i);
                } else {
                        return x & ~(1 << i);
                }
        }
        void bits_printf (unsigned int x) {
                int i;
                for ( i = 0; i < bits_length(x) ; i++ ) {
                        printf ("%d", bits_get(x,i));
                }
        }
        /*-------------------------driver.c---------------------------*/
        / *
        *       Driver program to test the bitstring package.
        * /
        #include <stdio.h>
        #include "bits.h"

9       int main (int argc, char* argv[]) {
                int i;   /* a generic counter */
                int x_in;
10              bits x, x_comp;

                        /* read in an integer */
11              printf ("Give me an integer:");
12              scanf ("%d", &x_in);
                        /* convert into a bit string */
13              x = bits_from_int(x_in);

                        /* print the bits of x */
                printf ("The %d bits of %d are:",
                                bits_length(x), x_in);
                bits_printf (x);
                printf ("\n");

                        /* complement the bits of x */
                printf ("The complement bits are:");
14              x_comp = x;         /* does this work always? */
15              for ( i = 0; i < bits_length(x); i++ ) {
                                /* is not something fishy here? */
                        x_comp = bits_put(x_comp, i, ~bits_get(x_comp, i));
                }
                bits_printf (x_comp);
                printf ("\n");
                return (0);
        }
```