Topic: Software Tools and the Basic Anatomy of C Programs

Approach: Dissect actual C programs

Overview and Intro:

Review the big picture: HTML, connector, script, tools
Demo: trainsched.html→trainsched.cgi→trainsched→tools
Need more tools to make it better

Crucial Concept: The Software Tool: Components and Connections

The Unix Philosophy
basic sequence: input, store, process, output
customization: activity can be directed by command line arguments

Facts about C Programs:

Most Unix tools are written in C
C is a compiled language
Program is written in a text editor (e.g. simple.c)
The *source code* is compiled into executable code
The executable code is run

Three Tools for the sched datafile

semi2tab1.c	filter to change semicolons to tabs
capitalize1.c	filter to upper case initials

We can now use them in pipelines and in our trainsched script

Looking at C Programs: The hello series

Computers do four things: input, storage, processing, output
Structure: Storage has structure, Code has structure

functions	a C program consists of one or more functions a function has: type, name, args, storage, code
storage	simple variables: int, char, float (easy conversion) variations: long, short, double, unsigned no Boolean: instead 0=false, non-0=non-false arrays: sequence of storage boxes: defining, using
operations	arithmetic(+,-,*,/), comparison, boolean
control flow	while, for, if
functions	defining, using, passing data, declaring
arguments	args to functions: simple passed by value, arrays by ref
input/output	numbers, characters, strings
strings	an array of chars terminated with a null char

More train sched Processing Tools:

empties.c	find fields with no data
missing.c	find lines missing one or more tags
outoforder.c	find lines with tags in wrong order

Good Programming Tip #1:

Use *gcc -Wall* to compile and fix each warning

```
:::::::::::::: trainsched ::::::::::::::
#!/bin/sh
# trainsched
#  usage: trainsched train#
# action: finds all entries for that TR and selects time and stn cols
#
       if test $# != 1
       then
              echo "usage: trainsched trainnum"
              exit 1
       fi
       grep "TR=$1;" sched | cut -d";" -f4,5 | sort -t= -k2n
:::::::::::::: trainsched.html ::::::::::::::
<!DOCTYPE html>
<html><head><title>Show Schedule for a Train</title></head>
<body>
<form action='trainsched.cgi' method='get'>
       Train number? <input type='text' name='trainnumber' size='10' />
       <input type='submit' value=' Show Schedule '  />
</form>
</body></html>
:::::::::::::: trainsched.cgi ::::::::::::::
#!/bin/sh
# trainsched.cgi
# connector for getting train schedule
#
       eval $(./qryparse)                  # get data from request
       echo "Content-type: text/plain"            # tell browser data type
       echo ""                             # end of header
       ./trainsched "$trainnumber"         # the content
:::::::::::::: semi2tab1.c ::::::::::::::
#include       <stdio.h>

/*
 * semi2tab1.c
 *   purpose: filter data replacing semicolons with tab chars
 *     input: text
 *    output: text with tabs in place of semicolons
 *    errors: no error conditions
 *     usage: semi2tab < input > output
 */
int main()
{
       int    c;             /* ask for storage for an integer value */
       while(1)                      // repeat until told to stop
       {
              c = getchar();        /* read in one char    */
              if ( c == EOF )              /* end of data?              */
                     break;         /* yes, get out of loop       */
              if ( c == ';' )
                     c = '\t';      /* replace            */
              putchar(c);           /* send to output     */
       }
       return 0;
}
:::::::::::::: semi2tab2.c ::::::::::::::
#include       <stdio.h>
/*
 * semi2tab2.c
 *   purpose: filter data replacing semicolons with tab chars
 *     input: text
 *    output: text with tabs in place of semicolons
 *    errors: no error conditions
 *     usage: semi2tab < input > output
 *     notes: version 2 uses the common compact C syntax for input loops
 */
int main()
{
       int    c;             // this is ok as a comment, too

       while( (c = getchar()) != EOF ){
              if ( c == ';' )
                     c = '\t';      /* replace            */
              putchar(c);           /* send to output     */
       }
       return 0;
}
```

```
:::::::::::::: capitalize1.c ::::::::::::::
#include       <stdio.h>
#include       <ctype.h>
/*
 * capitalize1.c
 *   purpose: filter data capitalizing initials
 *     input: text
 *    output: text with chars after '=' or ' ' changed to upper case
 *    errors: no error conditions
 *     usage: capitalize < input
 *      TODO: make it more general than just '=' or ' '
 */
int main()
{
       int     c;                              // data
       int     prevchar = '\0';                // the prev char

       while( ( c = getchar() ) != EOF )
       {
              if ( prevchar == '=' || prevchar == ' ' ){
                     if ( islower(c) ){
                            c = toupper(c);
                     }
              }
              putchar(c);            /* send to output      */
              prevchar = c;          /* record as prevchar  */
       }
       return 0;
}
:::::::::::::: count_semis1.c ::::::::::::::
#include       <stdio.h>

/*
 * count_semis1.c
 *      purpose: count the number of semicolons on each line
 *        usage: count_semis1 < data
 *       action: prints out the number of semicolons tab linenumber
 *       method: read chars, count semis; at newline print and reset count
 *         TODO: print the contents of the lines, not just the number
 */

int
main()
{
       char    c;
       int     line_num = 1;
       int     num_semis = 0;

       while( ( c = getchar() ) != EOF )
       {
              if ( c == '\n' ){
                     printf("%d\t%d\n", num_semis, line_num );
                     line_num++;
                     num_semis = 0;
              }
              else if ( c == ';' ){
                     num_semis++;
              }
       }
}
:::::::::::::: hello1.c ::::::::::::::

#include       <stdio.h>

/*
 *      purpose: print out a message and
 *               show the basic structure of a C program
 */

int main()
{
       printf( "Hello!\n" );         /* call the printf function */
       return 0;
}
```

```
:::::::::::::: hello2.c ::::::::::::::
#include       <stdio.h>

/*
 *  hello2.c
 *
 *      purpose: print out a message some number of times
 *               shows how the for loop works and numeric output
 */


int
main()
{
        int     i, maxnum;              /* counter and limit          */

        maxnum = 5 ;                    /* assign a value to maxnum   */

        for( i = 0 ; i<maxnum ; i = i + 1 )
                printf("%d/%d hello.\n", i, maxnum ); /* %d is plugin spot   */

        return 0;
}
:::::::::::::: hello3.c ::::::::::::::
#include       <stdio.h>

/*
 *  hello3.c
 *
 *      purpose: show how to input numbers and strings
 */

#define BUFLEN  100

int
main()
{
        int     i, maxnum = 0;          /* counter and limit          */
        char    message[BUFLEN];        /* an array of chars          */


        printf("Print what string? ");
        fgets( message, BUFLEN, stdin );/* read in a string, maxlen=100    */

        printf("Repeat it how many times? ");
        scanf( "%d", &maxnum );              /* read in a number           */

        for ( i = 0 ; i<maxnum ; i++ )
                printf( "%d/%d. %s", i, maxnum, message );   /* print it    */
        return 0;
}
```

```
:::::::::::::: hello4.c ::::::::::::::
#include       <stdio.h>

/*
 *  hello4.c
 *
 *      purpose: show how to create a function to do a subtask
 *               shows how to use #define for clarity and convenience
 */

#define STRSIZE        100
#define HOWMANYTIMES   3

void repeat_a_message( char [], int );       /* function declaration */

int
main()
{
       int     maxnum ;                      /* limit              */
       char    message[STRSIZE];     /* an array of chars          */

       printf("Print what string? ");
       fgets(message, STRSIZE, stdin);       /* read in a string           */

       printf("Repeat it how many times? ");
       scanf( "%d", &maxnum );               /* read in a number           */

       repeat_a_message( message, maxnum );
       repeat_a_message( "all done\n"  , HOWMANYTIMES );
       return 0;
}

void
repeat_a_message( char message[STRSIZE], int times )
/*
 * purpose: prints out the string in ‘message’ for ‘times’ iterations
 */
{
       int     i;                    /* local variable             */

       for ( i = 0 ; i<times ; i++ )
              printf( "%d. %s", i, message );       /* print it    */
}
:::::::::::::: hello5.c ::::::::::::::
#include       <stdio.h>

/*
 *  hello5.c
 *
 *      purpose: shows how to write a function that returns a value
 *               shows the while() loop, too, and the if() statement
 */

#define STRSIZE        100

void repeat_a_message(char [], int);
int  get_a_positive_number();

int
main()
{
       int     maxnum;               /* limit                      */
       char    message[STRSIZE];     /* an array of chars          */

       printf("Print what string? ");
       fgets(message, STRSIZE, stdin);       /* read in a string           */

       maxnum = get_a_positive_number();
       repeat_a_message( message, maxnum );

       return 0;
}
```

```
void repeat_a_message( char message[STRSIZE], int times )
/*
 * purpose: prints out the string in ‘message’ for ‘times’ iterations
 */
{
        int     i;                      /* local variable             */

        for ( i = 0 ; i<times ; i++ )
                printf( "%d. %s", i, message );       /* print it    */
}

int
get_a_positive_number()
/*
 * purpose: demand a positive integer from the user
 * returns: a positive integer
 */
{
        int     inputval = -1;

        while( inputval <= 0 )
        {
                printf("Repeat it how many times? ");
                scanf( "%d", &inputval );
                if ( inputval <= 0 )
                        printf("%d is not positive number\n", inputval);
        }
        return inputval;       /* send value back to caller */
}
:::::::::::::: hello6.c ::::::::::::::
#include       <stdio.h>
#include       <ctype.h>
#include       <stdlib.h>

/*
 *  hello6b.c
 *
 *      An enhanced version of hello6.c that checks for
 *      a number as being all digits OR a minus sign followed
 *      by all digits
 */

#define STRSIZE        100
#define YES 1
#define NO  0

void repeat_a_message(char [], int);
int  get_a_positive_number();

int
main()
{
        int     maxnum;                /* limit                       */
        char    message[STRSIZE];      /* an array of chars           */

        printf("Print what string? ");
        fgets(message, STRSIZE, stdin);       /* read in a string           */

        maxnum = get_a_positive_number();
        repeat_a_message( message, maxnum );

        return 0;
}

void
repeat_a_message( char message[STRSIZE], int times )
/*
 * purpose: prints out the string in ‘message’ for ‘times’ iterations
 */
{
        int     i;                      /* local variable             */

        for ( i = 0 ; i<times ; i++ )
                printf( "%d. %s", i, message );       /* print it    */
}

int
get_a_positive_number()
/*
 * purpose: demand a positive integer from the user
 * returns: a positive integer
 */
```

```
{
        int     inputval = -1;
        char    inputstr[STRSIZE];
        int     is_all_digits(char []);

        while( inputval <= 0 )
        {
                printf("Repeat it how many times? "); /* prompt      */
                fgets(inputstr, STRSIZE, stdin );     /* input       */
                if ( is_an_integer( inputstr ) == NO )
                        printf("This is not a number: %s", inputstr );
                else {
                        inputval = atoi( inputstr );
                        if ( inputval <= 0 )
                                printf("%d is not positive number\n", inputval);
                }
        }
        return inputval;       /* send value back to caller */
}
```

```
int
is_an_integer( char str[STRSIZE] )
/*
 * purpose: examine a string and see if all the chars are digits
 *          or if the first char is a '-' and the rest are digits
 * returns: 1 if all chars before the newline are digits, 0 otherwise
 * bug?:    what if a string with no chars appears?
 */
{
        int     pos;                    /* index into string   */

        pos = 0;
        if ( str[pos] == '-' )          /* leading '-' is OK   */
                pos++;
        while( str[pos] != '\n' ){              /* until done  */
                if ( ! isdigit((int)str[pos]) )       /* if not a dig       */
                        return NO;              /* get out now! */
                pos++;                          /* advance one */
        }
        return YES;                     /* no problems         */
}

:::::::::::::: hello7.c ::::::::::::::
#include       <stdio.h>
#include       <stdlib.h>
/*
 *  hello7.c
 *      purpose: show how to accept input from command line
 */

#define STRSIZE        100
void repeat_a_message(char [], int);

int
main(int argcount, char *arglist[])
{
        int     maxnum;                 /* limit                       */

        maxnum = atoi( arglist[2] );
        repeat_a_message( arglist[1], maxnum );

        return 0;
}

void
repeat_a_message( char message[STRSIZE], int times )
/*
 * purpose: prints out the string in 'message' for 'times' iterations
 */
{
        int     i;                      /* local variable              */

        for ( i = 0 ; i<times ; i++ )
                printf( "%d. %s\n", i, message );     /* print it    */
}
```

```
:::::::::::::: count_semis2.c ::::::::::::::
#include       <stdio.h>

/*
 * count_semis2.c
 *      purpose: count the number of semicolons on each line
 *        usage: count_semis < data
 *       action: prints out the number of semicolons tab line# tab line
 *       errors: potential overflow, but no security risk
 *       method: read in line, count semis, print result
 */

#define LINELEN 10000          /* long enough? */

int     count_semis(char []);

int
main()
{
        char    line[LINELEN];
        int     num_semis, line_num = 1;

        while( fgets(line, LINELEN, stdin) != NULL )
        {
                num_semis = count_semis(line);
                printf("%d\t%s", num_semis, line);
        }
        return 0;
}
/*
 * count_semis
 *   purpose: return the number of semicolons in the string s
 *       arg: a string
 *      rets: an int
 */
int count_semis(char s[])
{
        int     pos;
        int     num = 0;

        for(pos = 0; s[pos] != '\0'; pos++)
                if ( s[pos] == ';' )
                        num++;
        return num;
}
```