23 String structures

Using structures, one can create a safe way of handling strings. The following complete program illustrates an ‘append’ function which never exceeds string capacity; if necessary, the old string memory will be freed and a larger block allocated.

The program breaks up the input text into lines with a prescribed maximum length.

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

typedef struct { int capacity; char * contents; } STRING;

void decr ( char * x )
{
  while ( *x != '\0' )
    if ( *x == '\n' )
      *x = '\0';
    else
      ++x;
}

STRING * make_string ( )
{
  STRING * str = (STRING*) calloc(1,sizeof(STRING));
  char * contents = (char*) calloc(100, 1);
  str->capacity = 100;
  str->contents = contents;
  return str;
}

void append_word ( STRING * str, char * word )
{
  int newlen;
  if (strlen(str->contents) == 0)
    newlen = strlen(word);
  else
    newlen = strlen(str->contents) + 1 + strlen(word);

  if ( newlen >= str->capacity )
  {
    char * newcontents = (char*)calloc(newlen+100, 1);
    if ( strlen(str->contents) == 0 )
      snprintf(newcontents, newlen+1, "%s", word);
    else
```

```
      snprintf(newcontents, newlen+1, "%s %s", str->contents, word);

    free ( str->contents ); // not discussed in this module

    str->contents = newcontents;
    str->capacity = newlen + 100;
  }
  else
  {
    int len = strlen(str->contents);
    if ( len == 0 )
      snprintf( str->contents, newlen+1, "%s", word);
    else
      snprintf( &(str->contents[len]), strlen(word)+2, " %s", word );
  }
}

int main( int argc, char * argv[] )
{
  if ( argc != 2 )
  {
    fprintf(stderr,"%s requires one argument, line length; abort\n",
        argv[0]);
    return -1;
  }

  int line_length = atoi ( argv[1] );
  STRING * str[1000];
  char buffer[200];
  int maxindex = 0;

  str[0] = make_string();
  while ( fgets ( buffer, 200, stdin ) != NULL )
  {
    decr(buffer);
    int buflen = strlen ( buffer );
    int first_in_word = 0;
    while ( first_in_word < buflen )
    {
      while  ( buffer[first_in_word] == ' ' )
        ++ first_in_word;
      if ( first_in_word < buflen )
      {
        char word[200];
        int i = first_in_word;
        while ( buffer[i] != ' ' && buffer[i] != '\0' )
        {
          word[i-first_in_word] = buffer[i];
          ++i;
        }
        word[i-first_in_word] = '\0';

        if ( strlen ( str[maxindex]->contents ) + strlen(word) >= line_length )
        {
          ++maxindex;
          str[maxindex] = make_string();
        }
        append_word ( str[maxindex], word );
        first_in_word = i;
      }
    }
  }

  int i;
  for(i=0; i<=maxindex; ++i)
     printf("%s\n", str[i]->contents);

  return 0;
}

% gcc str_struc.c

% cat x
We know that you highly esteem the kind of Learning taught in those
Colleges, and that the Maintenance of our young Men, while with

% a.out < x
a.out requires one argument, line length; abort

% a.out 40 < x
We know that you highly esteem the kind
of Learning taught in those Colleges,
and that the Maintenance of our young
Men, while with
```