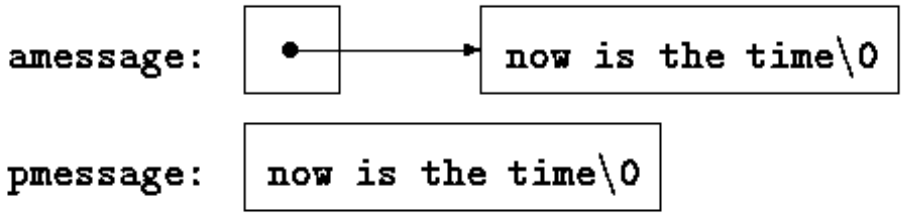
Character Arrays

```
strlen("hello, world"); /* string constant */
strlen(array); /* char array[100]; */
strlen(ptr); /* char *ptr; */

char *pmessage;
pmessage = "now is the time";
```

There is an important difference between these definitions:

```
char amessage[] = "now is the time"; /* an array */
char *pmessage = "now is the time"; /* a pointer */
```

amessage is an array, just big enough to hold the sequence of characters and '\0' that initializes it. Individual characters within the array may be changed but amessage will always refer to the same storage. On the other hand, pmessage is a pointer, initialized to point to a string constant; the pointer may subsequently be modified to point elsewhere, but the result is undefined if you try to modify the string contents.




```
/* strlen: return length of string s */
int strlen(char *s)
{
        int n;
        for (n = 0; s[n] != '\0', n++)
                ;
```

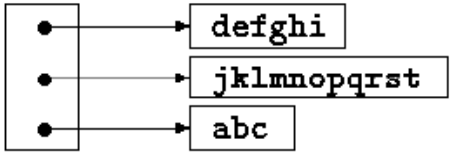
```
        return n;
}


/* strlen: return length of string s */
int strlen(char *s)
{
        int n;
        for (n = 0; *s != '\0', s++)
                n++;
        return n;
}

/* strcpy: copy t to s; array subscript version */
void strcpy(char *s, char *t)
{
        int i;
        i = 0;
        while ((s[i] = t[i]) != '\0')
                i++;
}
```

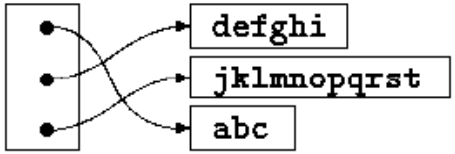
For contrast, here is a version of strcpy with pointers:

```
/* strcpy: copy t to s; pointer version */
void strcpy(char *s, char *t)
{
        while ((*s = *t) != '\0')
        {
```

```
                s++;
                t++;
        }
}

/* strcmp: return <0 if s<t, 0 if s==t, >0 if s>t */
int strcmp(char *s, char *t)
{
        int i;
        for (i = 0; s[i] == t[i]; i++)
                if (s[i] == '\0')
                        return 0;
        return s[i] − t[i];
}

/* strcmp: return <0 if s<t, 0 if s==t, >0 if s>t */
int strcmp(char *s, char *t)
{
        for ( ; *s == *t; s++, t++)
                if (*s == '\0')
                        return 0;
        return *s − *t;
}


/* strlen: return length of string s */
int strlen(char *s)
{
        char *p = s;
```

```
        while (*p != '\0')
                p++;
        return p − s;
}
```





```
char *pmessage;
pmessage = "now is the time";

char pmessage[MAXLINE];
sprintf(pmessage,”hello %d\n”,mynumber)

char pmessage[MAXLINE];
sscanf(”%d”,&mynumber);
sscanf(”%d”,pmessage)
```

(Standard) Input and Output

Input
Keyboard, file, network

Output

Screen, file, printer, network

Abstraction

```
int  scanf(        char *format, ...)
int fscanf(FILE *, char *format, ...)
int sscanf(char *, char *format, ...)

int  printf(        char *format, ...)
int fprintf(FILE *, char *format, ...)
int sprintf(char *, char *format, ...)



fp = fopen("TENLINES.TXT","w");
$ proj < myinputfile.txt

$ dir > myfilelist.txt

stdin, stdout, stderr

#include "stdio.h"
int
main( )
{
        FILE *funny;
        int c;
        funny = fopen("TENLINES.TXT","r");
        if (funny == NULL)
```

```
                printf("File doesn't exist\n");
        else
        {
                do
                {
                        c = getc(funny); /* get one character from the file */
                        putchar(c); /* display it on the monitor */
                } while (c != EOF); /* repeat until EOF (end of file) */
                fclose(funny);
        }
}

#include "stdio.h"
int
main( )
{
        FILE *fp1;
        char oneword[100];
        int c;
        fp1 = fopen("TENLINES.TXT","r");
        do
        {
                c = fscanf(fp1,"%s",oneword); /* got one word from the file */
                printf("%s\n",oneword); /* display it on the monitor */
        } while (c != EOF); /* repeat until EOF */
        fclose(fp1);
}

#include "stdio.h"
```

```
#define MAXLINE 1000
int
main( )
{
        FILE *fp1;
        char oneline[MAXLINE];
        char *c;
        fp1 = fopen("TENLINES.TXT","r");
        do
        {
                c = fgets(oneline, MAXLINE,fp1); /* get one line from the file */
                if (c != NULL);
                        printf("%s",oneline); /* display it on the monitor */
        } while (c != NULL); /* repeat until NULL */
        fclose(fp1);
}

#include <stdio.h>
#include <ctype.h>
main() /* lower: convert input to lower case*/
{
        int c
        while ((c = getchar()) != EOF)
                putchar(tolower(c));
        return 0;
}

void filecopy(FILE *ifp, FILE *ofp)
{
```

```
        int c;
        while ((c = getc(ifp)) != EOF)
                putc(c, ofp);
}

#include <stdio.h>
main()
{
        double sum, v;
        sum = 0;
        while (scanf("%lf", == 1)
                printf("\t%.2f\n", sum += v);
        return 0;
}

FILE *fp;
FILE *fopen(char *name, char *mode);
fp = fopen(name, mode)
read("r"), write ("w"), and append ("a").

if ((fp = fopen(myfilename, "r")) == NULL)
        ...

fprintf(stderr, "%s: error writing stdout\n", prog);

char *fgets(char *line, int maxline, FILE *fp)
int fputs(char *line, FILE *fp)
```

*char *fgets(char *line, int maxline, FILE *fp)*
fgets reads the next input line (including the newline) from file fp into the character array line; at most maxline−1 characters will be read. The resulting line is terminated with '\0'. Normally fgets returns line; on end of file or error it returns NULL. (Our getline returns the line length, which is a more useful value; zero means end of file.)

For output, the function fputs writes a string (which need not contain a newline) to a file:
*int fputs(char *line, FILE *fp)*
It returns EOF if an error occurs, and non−negative otherwise.

```
/* getline: read a line, return length */
int getline(char *line, int max)
{
        if (fgets(line, max, stdin) == NULL)
                return 0;
        return strlen(line);
}
```

```
...
while  (getline(line, MAXLINE) != 0)
{
        printf(“Line is |%s|\n”,line);

        line[strlen(line)] = ‘\0’;

        printf(“Line is |%s|\n”,line);
```

```
        sscanf(“%d”,&mynum);
        sum = sum + mynum;

        sscanf(“%d %d”,&mynum1, &mynum2);

}
```

```
int
main( )
{
        char name1[12],name2[12],mixed[25];
        char title[20];

        strcpy(name1,"Rosalinda");
        strcpy(name2,"Zeke");
        strcpy(title,"This is the title.");
        printf(" %s\n\n"title);
        printf("Name 1 is %s\n",name1);
        printf(Name 2 is %s\n",name2);
        if (strcmp(name1,name2)>0) /* return 1 if name1 > name2 */
                strcpy(mixed,name1);
        else
                strcpy(mixed,name2);
        printf("The biggest name alphabetically is %s\n",mixed);
        strcpy(mixed,name1);
        strcat(mixed," ");
        strcat(mixed,name2);
        printf("Both names are %s\n",mixed);
}
```