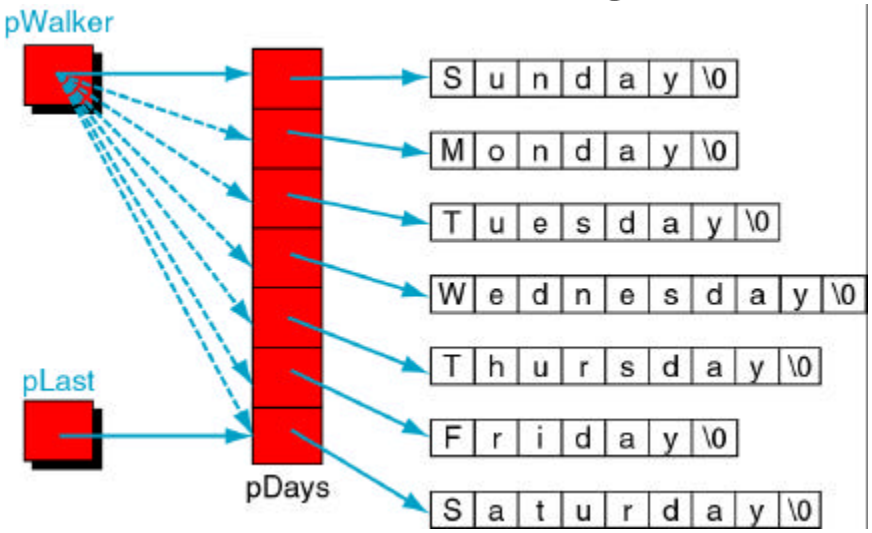
String Functions

COMP 1002/1402

Arrays of Strings

```
char *pDays[7];
pDays[0] ="Sunday";
pDays[0] ="Monday";
pDays[0] ="Tuesday";
pDays[0] ="Wednesday";
pDays[0] ="Thursday";
pDays[0] ="Friday";
pDays[0] ="Saturday";
```

Pointers to Strings



`gets` and **`fgets`**

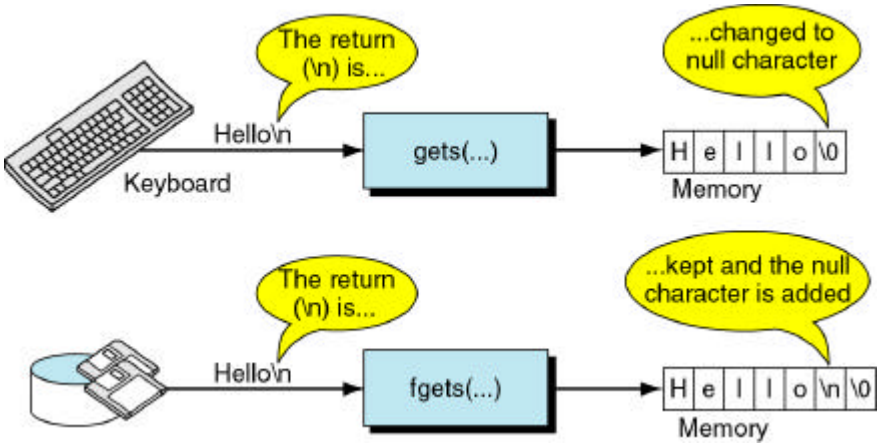
```
char *gets (char *strPtr);
char *fgets (char *strPtr,
    int size, FILE *fp);
```

If there is a problem it returns **`NULL`**

Otherwise it returns the pointer

Changes **`'\n'`** to **`'\0'`**

gets and **fgets**



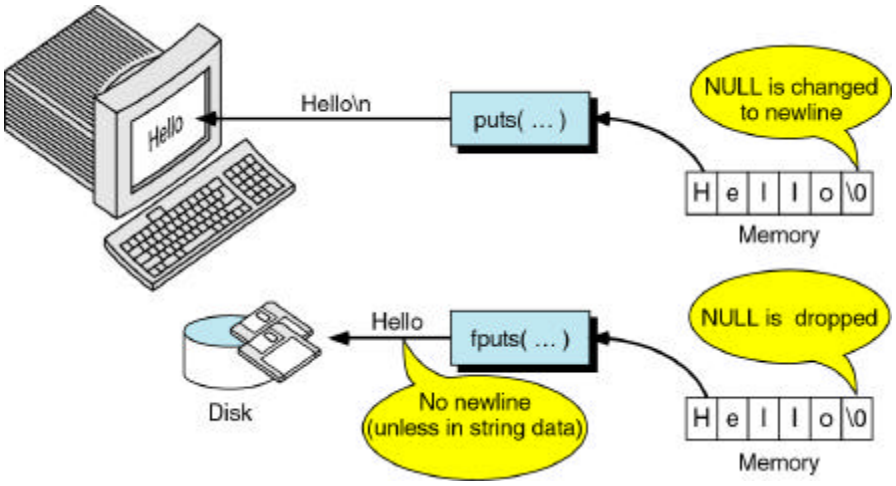
puts and **fputs**

```
int puts (const char *strPtr);
int fputs (const char *strPtr,
    FILE *fp);
```

If there is a problem it returns **EOF**

Outputs string may or may not have newline

puts and **fputs**



String Manipulation Functions

Variety of string functions

<string.h>

All begin with **str**

length

String Length (not including **'\0'**)

```
int strlen(const char *s);
```

Use:

```
char * pstr="Hello";
int i = strlen(pstr);
```

strlen

```
char strng[81];
…
fputs(strng, outFile);

if (strng[strlen(strng) – 1] != '\n'){
  fputs("\n", outFile);
}
```

String Copy (returns **`to`**)

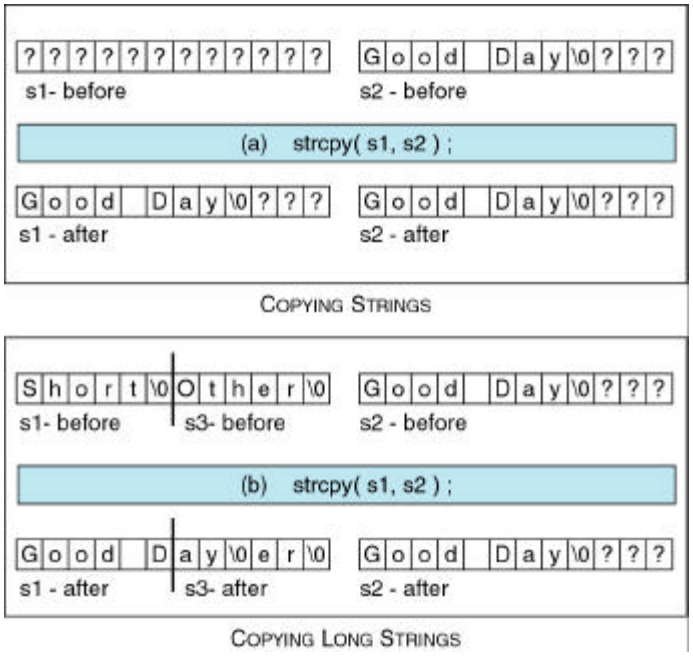
```
char * strcpy(char *to,
  const char *from);
```

`strcpy` – Not safe, length not specified

```
char * strcpy(char *to,
  const char *from, int size);
```

`strncpy` – Safer because length is specified

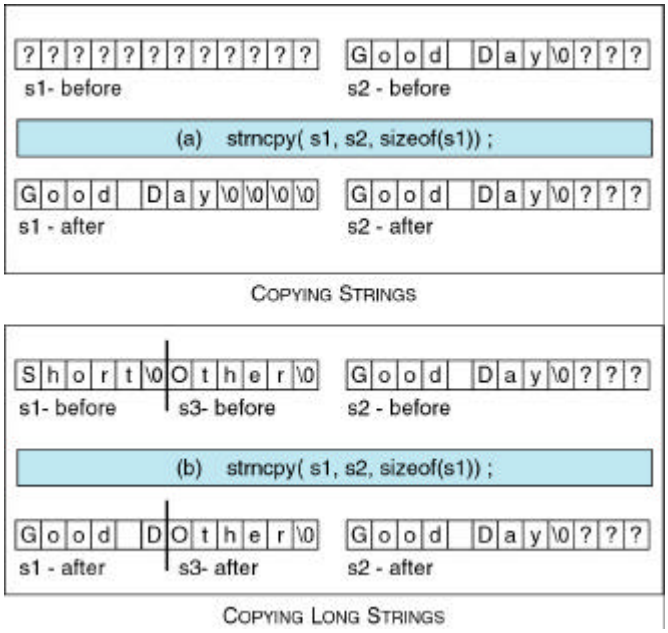
`strcpy`



COPYING STRINGS

COPYING LONG STRINGS

strncpy



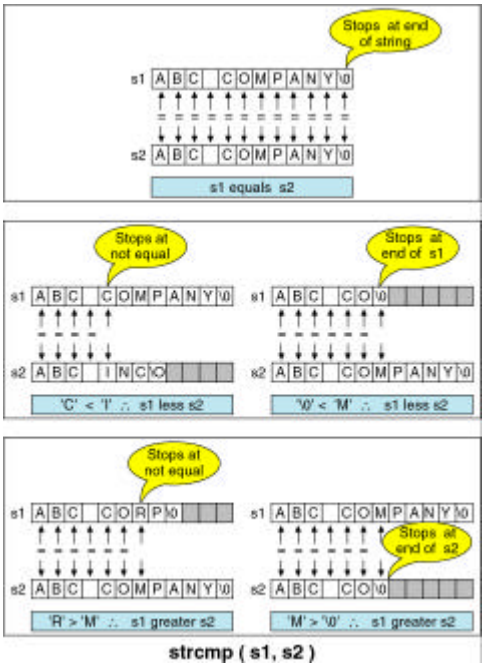
String Compare

```
int strcmp(const char *str1,
  const char *str2);
```

strcmp – Length not specified

```
int strncmp(const char *str1,
  const char *str2, int size);
```

strncmp – Compare up to specified length



strcmp (s1, s2)

strncmp(str1,str2,size)

str1	str2	Size	Results	Returns
"ABC123"	"ABC123"	8	equal	0
"ABC123"	"ABC456"	3	equal	0
"ABC123"	"ABC456"	4	str1 < str2	<0
"ABC123"	"ABC"	3	equal	0
"ABC123"	"ABC"	4	str1 < str2	>0
"ABC"	"ABC123"	3	equal	0
"ABC123"	"123ABC"	-1	equal	0

String Concatenate (returns **`to`**)

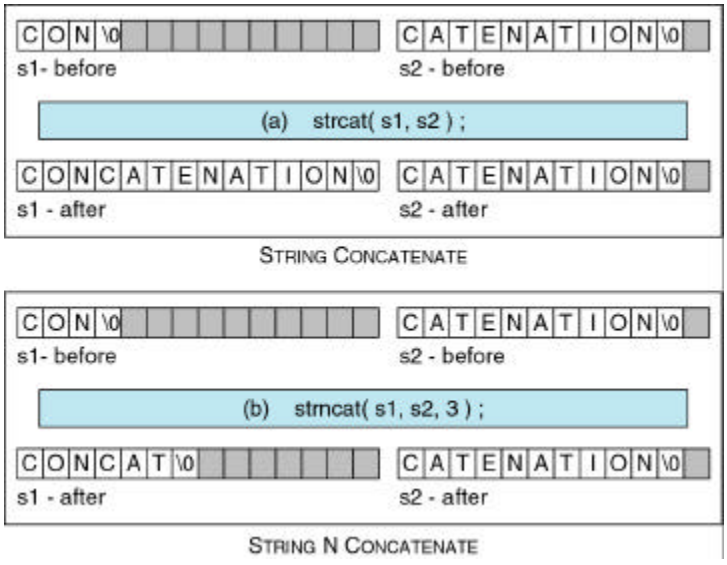
```
char * strcat(char *to, const
  char *from);
```

`strcat` – Not safe, length not specified

```
char * strcpy(char *to, const
  char *from, int size);
```

`strncat` – Safer because length is specified

`strcat` & **`strncat`**



Character in String

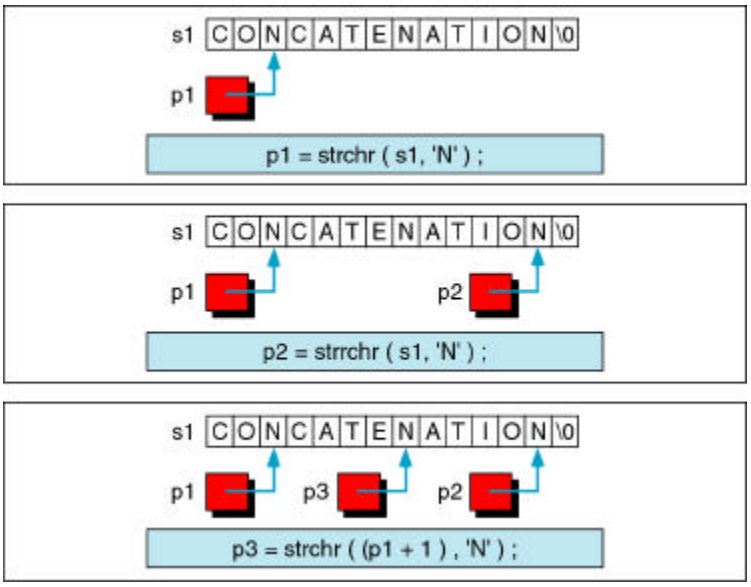
```
char *strchr(const char *string,
  int ch)
```

`strchr` – returns pointer to first match

```
char *strrchr(const char
  *string, int ch)
```

`strrchr` – returns pointer to last match

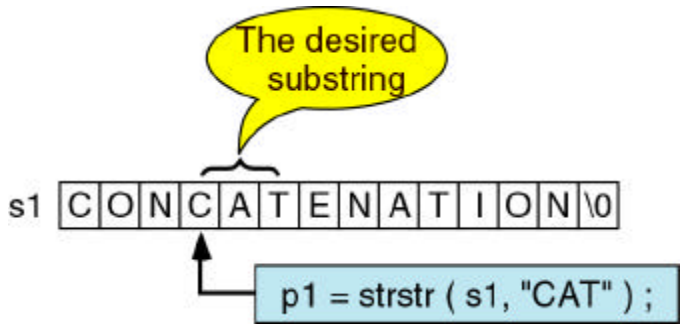
`strchr` & `strrchr`



String in String

```
char *strstr(const char *strng,
    const char *sub_str)
```

strstr – returns pointer to first match

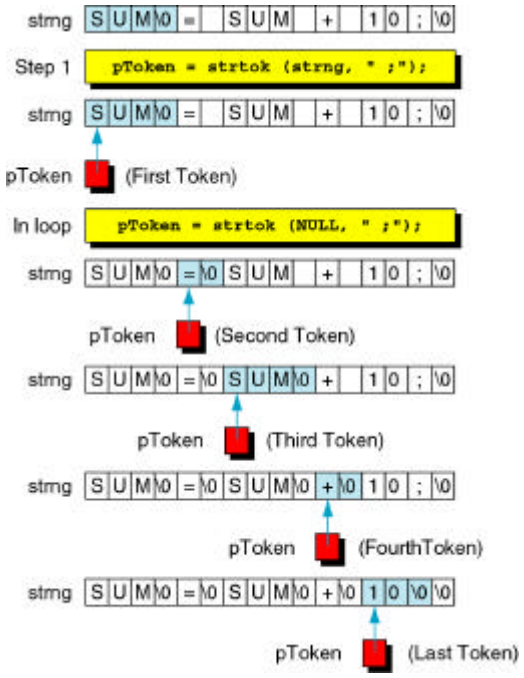


String Token

```
char *strtok(const char *strng,
    const char *delim)
```

strtok – returns pointer to first match after first delimiter

If **strng** is **NULL** then next pointer returned



Scan Memory String

```
int sscanf(char *str,
    const char *format,…);
```

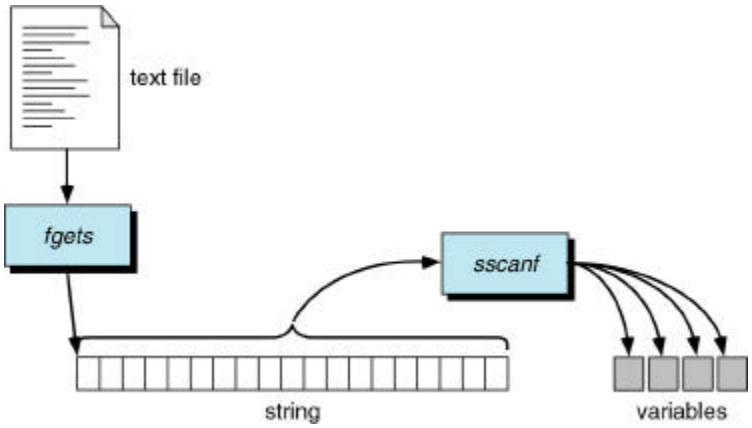
"reads" memory into variables

Uses format string of scanf...

Returns number of variables read

sscanf

Allows more control over data from file



Example

Input:

```
Einstein, Albert; 1234 67 D
```

Code:

```
i = sscanf(strIn,
  "%25[^;]%*c%4s%d%*[^ABCDF]%c",
  name, stuNo, &score, &grade);
```

Format Memory String

```
int sprintf(char *str,
    const char *format,…);
```

"prints" variables into memory

Uses format string of printf...

Returns **`EOF`** if error occurs

`sprintf`

Allows ease of use and control