strstr :

Usage : char *strstr(const char *haystack, const char *needle);

The strstr() function finds the first occurrence of the substring needle in the string haystack.

strchr:

Usage : char *strchr(const char *s, int c);

The strchr() function returns a pointer to the first occurrence of the character c in the string s.

strrchr:

Usage : char *strrchr(const char *s, int c);

The strchr() function returns a pointer to the last occurrence of the character c in the string s.

strtok:

Usage : char *strtok(char *str, const char *delim);

The function parses a string into a sequence of tokens. On the first call to strtok() the string to be parsed should be specified in str. Every subsequent call that should parse the same string, str should be NULL.

The delim argument specifies a set of characters that delimit the tokens in the parsed string. The caller may specify different strings in delim in successive calls that parse the same string.

Each call to strtok() returns a pointer to a null-terminated string containing the next token. This string does not include the delimiting character. If no more tokens are found, strtok() returns NULL.

strspn:

Usage : size_t strspn(const char *s, const char *accept);

The strspn() function calculates the length of the initial segment of s which consists entirely of characters in accept.

strcspn:

Usage : size_t strcspn(const char *s, const char *reject);

The strspn() function calculates the length of the initial segment of s which consists entirely of characters not in reject.

exit:

Usage : void exit(int status);

The exit() function causes normal process termination and the value of status & 0377 is returned to the parent

atexit:

Usage : int atexit(void (*function)(void));

The function registers the given function to be called at nor mal process termination, either via exit or via return from the program's main(). Functions so registered are called in the reverse order of their registration; no arguments are passed.

The same function may be registered multiple times: it is called once for each registration.

Example :

```
void f1() { static int i = 1 ; printf("in f1 for i=%d\n", i++) ; }
void f2() {printf("in f2\n") ;}
void f3() {printf("in f3\n") ;}
void f4() {printf("in f4\n") ;}

main() {
atexit(f1) ;   atexit(f2) ;   atexit(f1) ;   atexit(f1) ;   atexit(f3) ;   atexit(f4) ;
printf("processing ...\n") ;   sleep(1) ;   printf("processing ovr\n") ;
}
```