C Memoire

[<stdio.h> 2](#_Toc72967066)

[<stdlib.h> 3](#_Toc72967067)

[Library variables: 3](#_Toc72967068)

[Library macros 3](#_Toc72967069)

[Memory management 3](#_Toc72967070)

[Program termination 3](#_Toc72967071)

[Communicating with environment 3](#_Toc72967072)

[random 3](#_Toc72967073)

[Algorithms 3](#_Toc72967074)

[Example – get result of system command 4](#_Toc72967075)

# <stdio.h>

printf()

scanf(*const char \*restrict format, …*)

fscanf(*FILE \*restrict stream, const char \*restrict format, …*)

sscanf(*const char \*restrict buffer, const char \*restrict format, …*)

Returns number of arguments successfully assigned, or *EOF*, from *stdin* for *scanf()*, from *stream* for *fscanf()*, or from a null-terminated string *buffer* for *sscanf()*

fgets()

## printf formatting

# <stdlib.h>

## Library variables:

## Library macros

## Memory management

malloc()

calloc()

realloc()

free()

aligned\_alloc()

## Program termination

abort()

exit()

quick\_exit()

\_Exit()

atexit()

at\_quick\_exit()

EXIT\_SUCCESS

EXIT\_FAILURE

## Communicating with environment

system()

getenv()

getenv\_s()

popen()

## random

rand()

srand()

RAND\_MAX

## Algorithms

qsort()

qsort\_s()

bsearch()

bsearch\_s()

## Example – get result of system command

#include <stdlib.h>

#include <stdio.h>

#include <string.h>

#include <ctype.h>

char \*trim(char \*);

int main()

{

long CMD\_ALL\_MAX = 1024 \* 1024; // 1 MB

char cmd\_out\_all[CMD\_ALL\_MAX];

long CMD\_LINE\_MAX = 10 \* 1024; // 10 kB

char cmd\_out\_line[CMD\_LINE\_MAX];

FILE \*fp\_system\_cmd;

int status;

// command to use: list contents of $HOME

char \*cmd\_str = "ls ~";

//char \*cmd\_str = "cat /dev/random | head -c 1000";

// Execute cmd\_str with popen(), and open resulting stream

fp\_system\_cmd = popen(cmd\_str, "r");

if (fp\_system\_cmd == NULL) {

perror("fp\_system\_cmd=popen() error\n");

exit(2);

}

// Ongoing: 2021-05-27T00:11:21AEST If our goal is 'all output in single string', do we a loop with fgets()

// Read cmd\_str results line-by-line, appending to cmd\_out\_all

long \_strncat\_remaining;

while (fgets(cmd\_out\_line, CMD\_LINE\_MAX, fp\_system\_cmd) != NULL) {

\_strncat\_remaining = CMD\_ALL\_MAX - strlen(cmd\_out\_all);

strncat(cmd\_out\_all, cmd\_out\_line, \_strncat\_remaining);

//trim(cmd\_out\_line); // Definition/use of trim() ommited

printf("cmd\_out\_line=(%s)\n", cmd\_out\_line);

printf("strlen(cmd\_out\_line)=(%lu)\n", strlen(cmd\_out\_line));

}

printf("\n");

status = pclose(fp\_system\_cmd);

if (status == -1) {

perror("pclose error\n");

exit(2);

}

printf("cmd\_out\_all=(%s)\n", cmd\_out\_all);

printf("strlen(cmd\_out\_all)=(%lu)\n", strlen(cmd\_out\_all));

return 0;

}

## array\_size macro

// Works only for static arrays

#if !defined(ARRAY\_SIZE)

#define ARRAY\_SIZE(x) (sizeof((x)) / sizeof((x)[0]))

#endif