List of advanced programming tools in C language required for the "System programming" subject

(Excluding socket programming and OMP)

Header

stdio.h
stdlib.h
unistd.h
time.h
malloc.h
sys/stat.h
fcntl.h
dirent.h
pwd.h
signal.h

Directive

#include
#define
#if … #elif … #else … #endif

Variable

```
int argc;
char *argv;
char *envp;
FILE *stderr;
FILE *stdin;
FILE *stdout;
```

Named constant

EXIT_SUCCES	**S_IFREG**
RAND_MAX	**S_IFDIR**
SEEK_SET	**S_IFLNK**
SEEK_CUR	**S_IFBLK**
SEEK_END	**S_IFCHR**
S_IRUSR	**S_IFSOCK**
S_IWUSR	**EOF**
S_IXUSR	**L_tmpnam**
S_IRGRP	**SIGINT**
S_IWGRP	**SIGKILL**
S_IXGRP	**SIGTERM**
S_IROTH	**SIGSTOP**
S_IWOTH	**SIGCONT**
S_IXOTH	**SIGALRM**
O_RDONLY	**SIGUSR1**
O_WRONLY	**SIGUSR2**
O_RDWR	**SIGCHILD**
O_APPEND	**SIG_DFL**
O_TRUNC	**SIG_IGN**
O_CREAT	

Type, structure

time_t
FILE
DIR
pid_t

```
struct tm {
   int tm_sec;
   int tm_min;
   int tm_hour;
   int tm_mday;
   int tm_mon;
   int tm_year;
   int tm_wday;
   int tm_yday;
   };
```

```
struct dirent {
   int  d_ino;
   char d_name[256];
   };
```

```
struct stat {
   dev_t     st_dev;
   ino_t     st_ino;
   mode_t    st_mode;
   nlink_t   st_nlink;
   uid_t     st_uid;
   gid_t     st_gid;
   off_t     st_size;
   blksize_t st_blksize;
   blkcnt_t  st_blocks;
   time_t    st_atime;
   time_t    st_mtime;
   time_t    st_ctime;
   };
```

```
struct passwd {
   char *pw_name;
   uid_t pw_uid;
   gid_t pw_gid;
   char *pw_dir;
   char *pw_shell;
   };
```

Operator

All the 47 C operators must be known (including precedence and associativity).

Function, procedure

```
char *getenv(const char *name);
int putenv(char *string);
int system(const char *command);
time_t time(time_t *seconds);
char *ctime(const time_t *timer);
struct tm *localtime(const time_t *timer);
unsigned int sleep(unsigned int seconds);
int usleep(useconds_t usec);
void srand(unsigned int seed);
int rand(void);
void *malloc(size_t size);
void *calloc(size_t nitems, size_t size);
void *realloc(void *ptr, size_t size);
void *memset(void *str, int c, size_t n);
void free(void *ptr);
FILE *fopen(const char *filename, const char *mode);
int printf(const char *format, ...);
int fprintf(FILE *stream, const char *format, ...);
int sprintf(const char *str, const char *format, ...);
int fputs(const char *str , FILE *stream);
int scanf(const char *format, ...);
int fscanf(FILE *stream, const char *format, ...);
int sscanf(const char *str, const char *format, ...);
char *fgets(char *str, int n, FILE *stream);
int fseek(FILE *stream, long int offset, int whence)
int fflush(FILE *stream);
int feof(FILE *stream);
long int ftell(FILE *stream);
int fclose(FILE *stream);
int open(const char *pathname, int flags, mode_t mode);
ssize_t write(int fd, const void *buf, size_t count);
ssize_t read(int fd, void *buf, size_t count);
off_t lseek(int fd, off_t offset, int whence);
int close(int fd);
char *tmpnam(char *str);
FILE *tmpfile(void);
DIR *opendir(const char *dirname);
struct dirent *readdir(DIR *dirp);
int closedir(DIR *dirp);
int chdir(const char *path);
int stat(const char *restrict pathname, struct stat *restrict statbuf);
struct passwd *getpwuid(uid_t uid);
pid_t fork(void);
pid_t getpid(void);
pid_t getppid(void);
void signal(int sig, void (*func)(int));
int kill(pid_t pid, int sig);
int pause(void);
pid_t wait(int *wstatus);
unsigned int alarm(unsigned int seconds);
```

More details:

Linux manual pages:	https://man7.org/linux/man-pages/dir_section_3.html
Tutorialspoint:	https://www.tutorialspoint.com/c_standard_library
Header files:	https://pubs.opengroup.org/onlinepubs/009695399/basedefs/
C language:	https://www.programiz.com/c-programming