1. OPERATIONS ON FILES AND DESCRIPTORS

Include fcntl.h , unistd.h

int open(const char\* pathname, int flags, mode\_t mode);

int open(const char\* pathname, int flags);

flags defined in fcntl.h: O\_RDONLY, O\_WRONLY, O\_RDWR, O\_CREAT

ssize\_t read(int fd, void\* buf, size\_t count);

ssize\_t write(int fd, void\* buf, size\_t count);

int close(int fd);

int unlink(const char\* pathname);

1. CONCURRENT EXECUTION

Include unistd.h, sys/wait.h

pid\_t fork(void);

pid\_t wait(int \*status);

int pthread\_create(pthread\_t \*thread, NULL , void \*(\*start\_routine)(void\*), void \*arg);

void pthread\_exit(void \*value\_ptr);

int pthread\_join(pthread\_t thread, void \*\*value\_ptr);

int pthread\_detach(pthread\_t thread);

1. SEMAPHORES

Include semaphore.h

int sem\_init(sem\_t \*sem, 0 , unsigned int value);

int sem\_destroy(sem\_t \*sem);

int sem\_wait(sem\_t \*sem);

int sem\_post(sem\_t \*sem);

int sem\_getvalue(sem\_t \*sem, int \*sval);

sem\_t\* sem\_open(const char\* name, int oflag);

sem\_t\* sem\_open(const char\* name, int oflag, mode\_t mode, unsigned int value);

flags defined in fcntl.h

int sem\_close(sem\_t \*sem);

int sem\_unlink(const char \*name);

1. SOCKET

Include sys/socket.h

int socket(int domain, int type, 0 );

domain family defined in sys/socket.h, we only use AF\_INET family

we only use SOCK\_STREAM and SOCK\_DGRAM types

int connect(int socket, const struct sockaddr \*address, socklen\_t, address\_len);

int bind(int socket, const struct sockaddr \*address, socklen\_t, address\_len);

int listen(int sockfd, int backlog);

int accept(int socket, struct sockaddr \*address, socklen\_t \*address\_len);

ssize\_t send(int sockfd, const void \*buf, size\_t len, 0 );

ssize\_t recv(int sockfd, void \*buf, size\_t len, 0 );

ssize\_t sendto(int sockfd, const void \*buf, size\_t len, 0 , const struct sockaddr \*dest\_addr, socklen\_t addrlen);

ssize\_t recvfrom(int sockfd, void \*buf, size\_t len, 0 , struct sockaddr \*src\_addr, socklen\_t \*addrlen);

include arpa/inet.h

uint16\_t htons(uint16\_t hostshort);

uint16\_t ntohs(uint16\_t netshort);

in\_addr\_t inet\_addr(const char \*cp);

const char \*inet\_ntop(int af, const void \*src, char \*dst, socklen\_t size);

1. SHARED MEMEORY

Include sys/mman.h, sys/stat.h, sys/types.h, fcntl.h

int shm\_open(const char\* pathname, int flags, mode\_t mode);

int shm\_open(const char\* pathname, int flags);

flags defined in fcntl.h: O\_RDONLY, O\_WRONLY, O\_RDWR, O\_CREAT

int ftruncate(int fd, off\_t length);

void\* mmap(void \* addr, size\_t length, int prot, int flags, int fd, off\_t offset);

prot can be: PROT\_NONE, PROT\_EXAC, PROT\_READ, PROT\_WRITE,flag is MAP\_SHARED, offset is 0.

void\* munmap(void \* addr, size\_t length);

void\* shm\_unlink(const char\* pathname);

1. PIPE AND FIFO

Include unistd.h, sys.types.h, sys.stat.h

int pipe(int pipefd[2]);

int dup(int oldfd);

int dup2(int oldfd, int newfd);

int mkfifo(const char \*pathname, mode\_t mode);

to open, close, read and write of/on FIFO, refer to chapter 1

1. MEMORY AND STRINGS IN C

Include stdlib.h, string.h

void \*malloc(size\_t size);

void \*calloc(size\_t nmemb, size\_t size);

void \*realloc(void \*ptr, size\_t size);

void free(void \*ptr);

void \*memset(void \*s, int c, size\_t n);

void \*memcpy(void \*dest, const void\* src, size\_t n);

size\_t strlen(const char \*s);

int sprintf(char \*s, const char \*format, ...);

char \*strtok(char \*str, const char \*delim);

1. OPERATIONS ON FILES AND DESCRIPTORS

Include fcntl.h , unistd.h

int open(const char\* pathname, int flags, mode\_t mode);

int open(const char\* pathname, int flags);

flags defined in fcntl.h: O\_RDONLY, O\_WRONLY, O\_RDWR, O\_CREAT

ssize\_t read(int fd, void\* buf, size\_t count);

ssize\_t write(int fd, void\* buf, size\_t count);

int close(int fd);

int unlink(const char\* pathname);

1. CONCURRENT EXECUTION

Include unistd.h, sys/wait.h

pid\_t fork(void);

pid\_t wait(int \*status);

int pthread\_create(pthread\_t \*thread, NULL , void \*(\*start\_routine)(void\*), void \*arg);

void pthread\_exit(void \*value\_ptr);

int pthread\_join(pthread\_t thread, void \*\*value\_ptr);

int pthread\_detach(pthread\_t thread);

1. SEMAPHORES

Include semaphore.h

int sem\_init(sem\_t \*sem, 0 , unsigned int value);

int sem\_destroy(sem\_t \*sem);

int sem\_wait(sem\_t \*sem);

int sem\_post(sem\_t \*sem);

int sem\_getvalue(sem\_t \*sem, int \*sval);

sem\_t\* sem\_open(const char\* name, int oflag);

sem\_t\* sem\_open(const char\* name, int oflag, mode\_t mode, unsigned int value);

flags defined in fcntl.h

int sem\_close(sem\_t \*sem);

int sem\_unlink(const char \*name);

1. SOCKET

Include sys/socket.h

int socket(int domain, int type, 0 );

domain family defined in sys/socket.h, we only use AF\_INET family

we only use SOCK\_STREAM and SOCK\_DGRAM types

int connect(int socket, const struct sockaddr \*address, socklen\_t, address\_len);

int bind(int socket, const struct sockaddr \*address, socklen\_t, address\_len);

int listen(int sockfd, int backlog);

int accept(int socket, struct sockaddr \*address, socklen\_t \*address\_len);

ssize\_t send(int sockfd, const void \*buf, size\_t len, 0 );

ssize\_t recv(int sockfd, void \*buf, size\_t len, 0 );

ssize\_t sendto(int sockfd, const void \*buf, size\_t len, 0 , const struct sockaddr \*dest\_addr, socklen\_t addrlen);

ssize\_t recvfrom(int sockfd, void \*buf, size\_t len, 0 , struct sockaddr \*src\_addr, socklen\_t \*addrlen);

include arpa/inet.h

uint16\_t htons(uint16\_t hostshort);

uint16\_t ntohs(uint16\_t netshort);

in\_addr\_t inet\_addr(const char \*cp);

const char \*inet\_ntop(int af, const void \*src, char \*dst, socklen\_t size);

1. SHARED MEMEORY

Include sys/mman.h, sys/stat.h, sys/types.h, fcntl.h

int shm\_open(const char\* pathname, int flags, mode\_t mode);

int shm\_open(const char\* pathname, int flags);

flags defined in fcntl.h: O\_RDONLY, O\_WRONLY, O\_RDWR, O\_CREAT

int ftruncate(int fd, off\_t length);

void\* mmap(void \* addr, size\_t length, int prot, int flags, int fd, off\_t offset);

prot can be: PROT\_NONE, PROT\_EXAC, PROT\_READ, PROT\_WRITE,flag is MAP\_SHARED, offset is 0.

void\* munmap(void \* addr, size\_t length);

void\* shm\_unlink(const char\* pathname);

1. PIPE AND FIFO

Include unistd.h, sys.types.h, sys.stat.h

int pipe(int pipefd[2]);

int dup(int oldfd);

int dup2(int oldfd, int newfd);

int mkfifo(const char \*pathname, mode\_t mode);

to open, close, read and write of/on FIFO, refer to chapter 1

1. MEMORY AND STRINGS IN C

Include stdlib.h, string.h

void \*malloc(size\_t size);

void \*calloc(size\_t nmemb, size\_t size);

void \*realloc(void \*ptr, size\_t size);

void free(void \*ptr);

void \*memset(void \*s, int c, size\_t n);

void \*memcpy(void \*dest, const void\* src, size\_t n);

size\_t strlen(const char \*s);

int sprintf(char \*s, const char \*format, ...);

char \*strtok(char \*str, const char \*delim);