Antisèche :

NAME: read – read from a file descriptor
SYNOPSIS: ssize_t read(int fd, void *buf, size_t count);
DESCRIPTION: read() attempts to read up to count bytes from file descriptor fd into the buffer starting at buf.
RETURN VALUE: On success, the number of bytes read is returned (zero indicates end of file), and the file position is advanced by this number. It is not an error if this number is smaller than the number of bytes requested; this may happen for example because fewer bytes are actually available right now (maybe because we were close to end-of-file, or because we are reading from a pipe, or from a terminal), or because read() was interrupted by a signal. On error, −1 is returned, and errno is set appropriately.

NAME: write – write to a file descriptor
SYNOPSIS: ssize_t write(int fd, void *buf, size_t count);
DESCRIPTION: write() writes up to count bytes from the buffer starting at buf to the file referred to by the file descriptor fd.
RETURN VALUE: On success, the number of bytes written is returned (zero indicates nothing was written). It is not an error if this number is smaller than the number of bytes requested; this may happen for example because the disk device was filled. On error, −1 is returned, and errno is set appropriately.

NAME: file descriptor
NAME: read
read from a file descriptor
NAME: pipe
SYNOPSIS: int pipe(int pipefd[2]);
RETURN VALUE: On success, waitpid() returns the PID of the child whose state has changed; if WNOHANG was specified in the flags argument, waitpid() will return −1 and set errno to ECHILD. If all the children (or a process group) specified by pid exist, but have not yet changed state, then 0 is returned. On error, −1 is returned. The value of options is an OR of zero or more of the following constants: WNOHANG, WUNTRACED, WCONTINUED.
RETURN VALUE: On success, the PID of the child whose state has changed; if WNOHANG was specified and one or more children (ren) specified by pid exist, but have not yet changed state, then 0 is returned. On error, −1 is returned.

NAME: fork – create child process
SYNOPSIS: pid_t fork(void);
DESCRIPTION: fork() creates a new process by duplicating the calling process. The new process is referred to as the child process. The calling process is referred to as the parent process.
RETURN VALUE: On success, the PID of the child process is returned in the parent, and 0 is returned in the child. On failure, −1 is returned in the parent, no child process is created, and errno is set appropriately.

NAME: close – close a file descriptor
SYNOPSIS: int close(int fd);
DESCRIPTION: close() closes a file descriptor (for a regular file, a pipe or a socket), so that it no longer refers to any file.
RETURN VALUE: close() returns zero on success. On error, −1 is returned, and errno is set appropriately.

NAME: recv – receive a message from a socket
SYNOPSIS: ssize_t recv(int sockfd, void *buf, size_t len, int flags);
DESCRIPTION: The recv() call is used to receive messages from a socket. It may be used to receive data on connection-oriented sockets.
The only difference between recv() and read() is the presence of flags. With a 0 flags argument, recv() is generally equivalent to read().

NAME: send – send a message on a socket
SYNOPSIS: ssize_t send(int sockfd, const void *buf, size_t len, int flags);
DESCRIPTION: The send() system call sends() to transmit a message to another socket. The send() call may be used only when the socket is in a connected state (so that the intended recipient is known). The only difference between send() and write() is the presence of flags. With a 0 flags argument, send() is equivalent to write().

NAME: waitpid – wait for process to change state
SYNOPSIS: pid_t waitpid(pid_t pid, int *wstatus, int options);
DESCRIPTION: waitpid() is used to wait for state changes in a child of the calling process. A state change is considered to be: the child terminated; the child was stopped by a signal; or the child was resumed by a signal. In the case of a terminated child, performing a wait allows the system to release the resources associated with the child; if a wait is not performed, then the terminated child remains in a "zombie" state. If pid > 0, then the wait will wait for the children whose PID equals pid. If wstatus is not NULL, then waitpid() stores status in informations in the int it points to. If wstatus is NULL, then this parameter is ignored.
The value of options is an OR of zero or more of the following constants: WNOHANG, WUNTRACED, WCONTINUED.
RETURN VALUE: On success, waitpid() returns the PID of the child whose state has changed; if WNOHANG was specified and one or more child(ren) specified by pid exist, but have not yet changed state, then 0 is returned. On error, −1 is returned.

NAME: pipe – create pipe
SYNOPSIS: int pipe(int pipefd[2]);
DESCRIPTION: pipe() creates a pipe, a unidirectional data channel that can be used for interprocess communication. The array pipefd is used to return two file descriptors, referring to the ends of the pipe. pipefd[0] refers to the read end, pipefd[1] refers to the write end. Data written to the write end of the pipe is buffered by the kernel until it is read from the read end of the pipe.

If all file descriptors referring to the write end of a pipe have been closed, then an attempt to read from the pipe will see end-of-file and will return 0. If all file descriptors referring to the read end of a pipe have been closed, then a write will cause a SIGPIPE signal to be generated for the calling process. If the calling process is ignoring this signal, then write fails with the error EPIPE. An application that uses pipe and fork should use suitable close calls to close unnecessary duplicate file descriptors; this ensures that end-of-file and SIGPIPE/EPIPE are delivered when appropriate.
RETURN VALUE: On success, zero is returned. On error, −1 is returned, and errno is set appropriately.

EN–TETE : int create_server_socket(const char *port);
DESCRIPTION : Il s'agit d'un fonction de la bibliothèque socklib, vue en CM/TD/TP. Retourne une socket côté serveur, à écouter sur le port passé en paramètre.
RETOUR : Retourne le socket d'écoute en cas de succès, −1 en cas d'échec.

EN–TETE : int accept_connection(int s);
DESCRIPTION : Il s'agit d'un fonction de la bibliothèque socklib, vue en CM/TD/TP. Se met en attente bloquante d'une connexion sur la socket d'écoute s. Lorsqu'un client vient se connecter, retourne la socket de dialogue créée.
RETOUR : Retourne le socket de dialogue en cas de succès, −1 en cas d'échec.