Fonctions C Essentielles pour ft_ping

Bibliothèques Nécessaires

```c
#include <sys/socket.h>   // Sockets, sendto, recvfrom
#include <netinet/ip.h>   // Structure IP
#include <netinet/ip_icmp.h> // Structure ICMP
#include <netdb.h>        // getaddrinfo, gethostbyname
#include <arpa/inet.h>    // inet_ntoa, htons, ntohs
#include <signal.h>       // sigaction, SIGINT
#include <getopt.h>       // getopt_long
#include <sys/time.h>     // gettimeofday
#include <time.h>         // clock_gettime
#include <unistd.h>       // getpid, close
#include <errno.h>        // errno, strerror
#include <string.h>       // memset, memcpy, strlen
#include <stdio.h>        // printf family
#include <stdlib.h>       // exit, malloc, free
```

Fonctions de Parsing et Arguments

Fonction	Prototype	Retour	Description	Pourquoi Essentielle
`getopt_long`	`int getopt_long(int argc, char **argv, const char *optstring, const struct option *longopts, int *longindex)`	Option trouvée ou -1	Parse les arguments courts (-v) et longs (--ttl)	Gestion propre des options comme --ttl, -w, -s
`optarg`	`extern char *optarg`	Pointeur vers argument	Contient la valeur de l'option courante	Récupère "5" dans "--ttl 5"
`optind`	`extern int optind`	Index	Position actuelle dans argv	Trouve l'adresse cible après les options

Fonctions Réseau - Résolution DNS

Fonction	Prototype	Retour	Description	Pourquoi Essentielle
`getaddrinfo`	`int getaddrinfo(const char *node, const char *service, const struct addrinfo *hints, struct addrinfo **res)`	0 si succès	Résout nom → IP (moderne)	Convertit "google.com" en 142.250.185.14
`freeaddrinfo`	`void freeaddrinfo(struct addrinfo *res)`	void	Libère mémoire getaddrinfo	Évite les fuites mémoire
`gai_strerror`	`const char *gai_strerror(int errcode)`	Message d'erreur	Messages d'erreur getaddrinfo	Debug des erreurs DNS
`inet_ntoa`	`char *inet_ntoa(struct in_addr in)`	String IP	Convertit IP binaire → string	Affichage des adresses dans les logs

Fonctions Socket Raw - Le Cœur du Projet

Fonction	Prototype	Retour	Description	Pourquoi Essentielle
`socket`	`int socket(int domain, int type, int protocol)`	File descriptor ou -1	Crée socket raw ICMP	`socket(AF_INET, SOCK_RAW, IPPROTO_ICMP)`
`setsockopt`	`int setsockopt(int sockfd, int level, int optname, const void *optval, socklen_t optlen)`	0 si succès	Configure options socket	Définir TTL avec IP_TTL
`sendto`	`ssize_t sendto(int sockfd, const void *buf, size_t len, int flags, const struct sockaddr *dest_addr, socklen_t addrlen)`	Octets envoyés ou -1	Envoie paquet ICMP	Envoie Echo Request
`recvfrom`	`ssize_t recvfrom(int sockfd, void *buf, size_t len, int flags, struct sockaddr *src_addr, socklen_t *addrlen)`	Octets reçus ou -1	Reçoit paquet ICMP	Reçoit Echo Reply + erreurs ICMP
`close`	`int close(int fd)`	0 si succès	Ferme socket	Nettoyage propre

Fonctions Timing - Mesure RTT

Fonction	Prototype	Retour	Description	Pourquoi Essentielle
clock_gettime	int clock_gettime(clockid_t clk_id, struct timespec *tp)	0 si succès	Temps haute précision	CLOCK_MONOTONIC pour RTT précis
gettimeofday	int gettimeofday(struct timeval *tv, struct timezone *tz)	0 si succès	Temps avec microsecondes	Alternative pour mesure RTT

Fonctions de Contrôle - Select/Timeout

Fonction	Prototype	Retour	Description	Pourquoi Essentielle
select	int select(int nfds, fd_set *readfds, fd_set *writefds, fd_set *exceptfds, struct timeval *timeout)	Nb FD prêts ou -1	Attend réponse avec timeout	Gère timeout de réception
FD_ZERO	void FD_ZERO(fd_set *set)	void	Initialise fd_set	Prépare select()
FD_SET	void FD_SET(int fd, fd_set *set)	void	Ajoute FD au set	Surveille socket ICMP

Fonctions Signal - Gestion Ctrl+C

Fonction	Prototype	Retour	Description	Pourquoi Essentielle
sigaction	int sigaction(int signum, const struct sigaction *act, struct sigaction *oldact)	0 si succès	Install gestionnaire signal	Intercepte SIGINT (Ctrl+C)
sigemptyset	int sigemptyset(sigset_t *set)	0 si succès	Vide masque signaux	Prépare sigaction

Fonctions Système - Utilitaires

Fonction	Prototype	Retour	Description	Pourquoi Essentielle
getpid	pid_t getpid(void)	PID du processus	ID unique processus	Identifiant dans paquets ICMP
strerror	char *strerror(int errnum)	Message d'erreur	Convertit errno → string	Messages d'erreur compréhensibles
sleep	unsigned int sleep(unsigned int seconds)	Temps restant	Pause en secondes	Intervalle entre pings
usleep	int usleep(useconds_t usec)	0 si succès	Pause en microsecondes	Timing précis

Fonctions Mémoire et String

Fonction	Prototype	Retour	Description	Pourquoi Essentielle
memset	void *memset(void *s, int c, size_t n)	Pointeur s	Remplit mémoire	Initialiser structures à zéro
memcpy	void *memcpy(void *dest, const void *src, size_t n)	Pointeur dest	Copie mémoire	Construire paquets ICMP
malloc	void *malloc(size_t size)	Pointeur ou NULL	Alloue mémoire	Buffers dynamiques si besoin
free	void free(void *ptr)	void	Libère mémoire	Évite fuites mémoire

Fonctions Printf - Affichage

Fonction	Prototype	Retour	Description	Pourquoi Essentielle
printf	int printf(const char *format, ...)	Nb caractères	Affichage standard	Messages principaux
fprintf	int fprintf(FILE *stream, const char *format, ...)	Nb caractères	Affichage sur flux	Erreurs sur stderr
snprintf	int snprintf(char *str, size_t size, const char *format, ...)	Nb caractères	Printf dans buffer	Construction de chaînes

Conversion Réseau - Endianness

Fonction	Prototype	Retour	Description	Pourquoi Essentielle
htons	uint16_t htons(uint16_t hostshort)	Valeur réseau	Host → Network (16 bits)	Ports et identifiants ICMP
ntohs	uint16_t ntohs(uint16_t netshort)	Valeur host	Network → Host (16 bits)	Lecture paquets reçus
htonl	uint32_t htonl(uint32_t hostlong)	Valeur réseau	Host → Network (32 bits)	Adresses IP
ntohl	uint32_t ntohl(uint32_t netlong)	Valeur host	Network → Host (32 bits)	Lecture adresses IP