Funkcje gniazd sieciowych

NAZWA	PROTOTYP	PARAMETRY I PRZYKŁADY	OPIS / UWAGI	ZWRACA / BŁĘDY
SOCKET	int socket(int family, int type, int protocol);	family: AF_INET (IPv4), AF_INET6 (IPv6), AF_IPX, AF_LOCAL/AF_UNIX, AF_PPPOX, AF_ROUTE, AF_NETLINK, AF_KEY, PF_* (aliasy) type: SOCK_STREAM, SOCK_DGRAM, SOCK_RAW, SOCK_PACKET (wycofane) protocol: 0 (domyślny), IPPROTO_SCTP, IPPROTO_ICMP	Tworzy gniazdo w jądrze systemu; deskryptor używany przez inne funkcje.	Zwraca deskryptor lub -1 przy błędzie (errno: EACCES, EINVAL, ENOBUFS itd.)
BIND	int bind(int sockfd, const struct sockaddr *addr, socklen_t addrlen);	addr: np. sockaddr_in z IP i portem, addrlen = sizeof(sockaddr_in)	Wiąże gniazdo z lokalnym adresem i portem, wymagane przed listen().	0 przy sukcesie, -1 przy błędzie.
LISTEN	int listen(int sockfd, int backlog);	backlog: rozmiar kolejki oczekujących połączeń.	Ustawia gniazdo w tryb nasłuchiwania (serwer TCP).	0 przy sukcesie, -1 przy błędzie.
ACCEPT	int accept(int sockfd, struct sockaddr *addr, socklen_t *addrlen);	addr: struktura z adresem klienta, addrlen: długość adresu.	Akceptuje połączenie przychodzące i tworzy nowe gniazdo dla klienta.	Nowy deskryptor połączenia lub -1 przy błędzie.
CONNECT	int connect(int sockfd, const struct sockaddr *addr, socklen_t addrlen);	addr: adres serwera (IPv4/IPv6).	Nawiązuje połączenie z serwerem TCP.	0 przy sukcesie, -1 przy błędzie.
SEND	ssize_t send(int sockfd, const void *buf, size_t len, int flags);	flags: MSG_DONTWAIT, MSG_MORE.	Wysyła dane przez połączenie TCP.	Liczba wysłanych bajtów lub -1 przy błędzie.
RECV	ssize_t recv(int sockfd, void *buf, size_t len, int flags);	flags: MSG_WAITALL, MSG_PEEK, MSG_DONTWAIT.	Odbiera dane z połączenia TCP.	0 (zamknięcie), liczba bajtów, lub -1 przy błędzie.
SENDTO	ssize_t sendto(int sockfd, const void *buf, size_t len, int flags, const struct sockaddr *dest_addr, socklen_t addrlen);	flags: MSG_MORE, MSG_DONTWAIT; dest_addr: adres odbiorcy UDP.	Wysyła pojedynczy datagram UDP.	Liczba bajtów lub -1 przy błędzie.
RECVFROM	ssize_t recvfrom(int sockfd, void *buf, size_t len, int flags, struct sockaddr *src_addr, socklen_t *addrlen);	flags: MSG_TRUNC, MSG_PEEK, MSG_DONTWAIT; src_addr: adres nadawcy.	Odbiera pojedynczy datagram UDP.	Liczba bajtów lub -1 przy błędzie.

Funkcje konwersji i adresowania

NAZWA	PROTOTYP	PARAMETRY I PRZYKŁADY	OPIS / UWAGI	ZWRACA
HTONS / HTONL	uint16_t htons(uint16_t v); uint32_t htonl(uint32_t v);	Konwersja Host→Network (big-endian).	Stosowana dla portów (htons) i pól 32-bit (htonl).	Wartość przekonwertowana.
NTOHS / NTOHL	uint16_t ntohs(uint16_t v); uint32_t ntohl(uint32_t v);	Konwersja Network→Host.	Symetryczne do powyższych.	Wartość przekonwertowana.
INET_PTON	int inet_pton(int family, const char *src,	family: AF_INET/AF_INET6, src:	Konwertuje adres IP (tekst)	1 przy sukcesie, 0 lub -1 przy

	void *dst);	'192.168.1.1'.	na binarny.	błędzie.
INET_NTOP	const char *inet_ntop(int family, const void *src, char *dst, socklen_t size);	family: AF_INET/AF_INET6, dst: bufor tekstowy.	Konwertuje adres IP binarny na tekstowy.	Wskaźnik dst lub NULL przy błędzie.

Struktury adresowe

STRUKTURA	KLUCZOWE POLA	OPIS
SOCKADDR_IN	sa_family_t sin_family; in_port_t sin_port; struct in_addr sin_addr;	Adres IPv4 i port. Używana w bind(), connect().
SOCKADDR_IN6	sa_family_t sin6_family; in_port_t sin6_port; struct in6_addr sin6_addr;	Adres IPv6 i port. Analogiczna do sockaddr_in.
ADDRINFO	ai_family, ai_socktype, ai_protocol, ai_addr, ai_next;	Uniwersalna struktura adresowa dla IPv4/IPv6.
IN_ADDR / IN6_ADDR	uint32_t s_addr; / struct in6_addr {...};	Reprezentacja binarna adresu IPv4/IPv6.