The function is defined for protocol-independent nodename-to-address translation. It performs functionality of and in more sophisticated manner. The addrinfo structure is defined as a result of including the header: struct addrinfo {
*

```
int     ai_flags;     /* AI_PASSIVE, AI_CANONNAME, AI_NUMERICHOST */
int     ai_family;    /* PF_xxx */
int     ai_socktype;  /* SOCK_xxx */
int     ai_protocol;  /* 0 or IPPROTO_xxx for IPv4 and IPv6 */
size_t  ai_addrlen;   /* length of ai_addr */
char    *ai_canonname; /* canonical name for nodename */
struct sockaddr  *ai_addr; /* binary address */
struct addrinfo  *ai_next; /* next structure in linked list */ };
```

The and arguments are pointers to null-terminated strings or One or both of these two arguments must be a pointer. In the normal client scenario, both the and are specified. In the normal server scenario, only the is specified. A string can be either a node name or a numeric host address string i.e., a dotted-decimal IPv4 address or an IPv6 hex address A string can be either a service name or a decimal port number. The caller can optionally pass an structure, pointed to by the third argument, to provide hints concerning the type of socket that the caller supports. In this structure all members other than and must be zero or a pointer. A value of for means the caller will accept any protocol family. A value of 0 for means the caller will accept any socket type. A value of 0 for means the caller will accept any protocol. For example, if the caller handles only TCP and not UDP, then the member of the hints structure should be set to when is called. If the caller handles only IPv4 and not IPv6, then the member of the structure should be set to when is called. If the third argument to is a pointer, this is the same as if the caller had filled in an structure initialized to zero with set to PF_UNSPEC. Upon successful return a pointer to a linked list of one or more structures is returned through the final argument. The caller can process each structure in this list by following the pointer, until a pointer is encountered. In each returned structure the three members and are the corresponding arguments for a call to the function. In each structure the member points to a filled-in socket address structure whose length is specified by the member. If the bit is set in the member of the structure, then the caller plans to use the returned socket address structure in a call to In this case, if the argument is a pointer, then the IP address portion of the socket address structure will be set to for an IPv4 address or for an IPv6 address. If the bit is not set in the member of the structure, then the returned socket address structure will be ready for a call to or either In this case, if the argument is a pointer, then the IP address portion of the socket address structure will be set to the loopback address. If the bit is set in the member of the structure, then upon successful return the member of the first structure in the linked list will point to a null-terminated string containing the canonical name of the specified If the bit is set in the member of the structure, then a string must be a numeric host address string. Otherwise an error of is returned. This flag prevents any type of name resolution service (e.g., the DNS) from being called. All of the information returned by is dynamically allocated: the structures, and the socket address structures and canonical node name strings pointed to by the addrinfo structures. To return this information to the system the function Fn freeaddrinfo is called. The structure pointed to by the is freed, along with any dynamic storage pointed to by the structure. This operation is repeated until a pointer is encountered. To aid applications in printing error messages based on the codes returned by is defined. The argument is one of the values defined earlier and the return value points to a string describing the error. If the argument is not one of the values, the function still returns a pointer to a string whose contents indicate an unknown error. Error return status from is zero on success and non-zero on errors. Non-zero error codes are defined in and as follows: address family for nodename not supported temporary failure in name resolution invalid value for ai_flags non-recoverable failure in name resolution ai_family not supported memory allocation failure no address associated with nodename nodename nor servname provided, or not known servname not supported for ai_socktype ai_socktype not supported system error returned in errno If called with proper argument, returns a pointer to a string describing the given error code. If the argument is not one of the values, the function still returns a pointer to a string whose contents indicate an unknown error. R. Gilligan, S. Thomson, J. Bound, and W. Stevens, ‘‘Basic Socket Interface Extensions for IPv6,’’ RFC2133, April 1997. The implementation first appeared in WIDE Hydrangea IPv6 protocol stack kit. The function is defined IEEE POSIX 1003.1g draft specification, and documented in ‘‘Basic Socket Interface Extensions for IPv6’’ The text was shamelessly copied from RFC2133.