ساختمان داده های مورد نیاز در برنامه نویسی مبتنی بر سوکت

- اولین نوع داده ، " مشخصه سوکت " یا Socket Descriptor است ،که برای ارجاع به یک ارتباط باز مورد استفاده قرار میگیرد مشخصه سوکت یک عدد صحیح دوبایتی است

```
int a ;
```

- دومین نوع داده برای ایجاد سوکت، یک استراکچراست که آدرس پورت پروسه و همچین آدرس IP ماشین طرف ارتباط را در خود نگه میدارد

```
Struct sockaddr {
    Unsigned short      sa_family ;      // Address family
    Char                sa_data[14] ;    //14 byte of protocol address
} ;
```

- **sa_family** این فیلد خانواده یا نوع سوکت را مشخص میکند در حقیقت این گزینه تعیین میکند سوکت مورد نظررا درچه شبکه ای و روی چه پروتکلی بکار خواهید گرفت، بنابراین در سیستمی با پروتکل و سوکت های متفاوت نوع سوکت باید تعیین شود در این فصل شبکه اینترنت با پروتکل TCP/IP است و خانواده سوکت را با ثابت AF_INET مشخص می کنیم.

- **sa_data** این چهارده بایت مجموعه ای است از آدرس پورت، آدرس IP و قسمتی اضافی که باید با صفر پر شود

((در شبکه اینترنت آدرس IP چهاربایتی و آدرس پورت دو بایتی است))

```
Struct sockaddr_ in     {
    short  int              sin_family ;        // Address family
unsigned  short  int        sin_port ;          // port number
    Struct  in_addr         sin_addr ;          // Internet address
     unsigned char          sin_zero [ 8 ] ;    // same size az struct sochaddr
  } ;
```

sin_family همانند ساختار قبلی خانواده سوکت را تعیین میکند و برای شبکه اینترنت مقدار ثابت AF _INET است

sin_port این فیلد دو بایتی،بایتی آدرس پورت پروسه مورد نظراست

sin_zero [8] این هشت بایت در کاربردهای مهندسی اینترنت کلا باید مقدار صفر داشته باشد زیرا در شبکه اینترنت فعلاً آدرس IP چهار بایتی و آدرس پورت دو بایتی است در حالی که در برخی دیگر از شبکه ها طول آدرس بیشتر است بنابراین جهت استفاده این استراکچر در شبکه اینترنت این هشت بایت اضافی باید با تابعی مانند memset() تماماً صفر شود

Sin_addr

آدرس IP ماشین مورد نظر را مشخص میکند این فیلد خود نیز یک استراکچر است که در حالت کلی عددی صحیح، بدون علامت و چهاربایتی است

```
// Internet address      (a structure  for  historical  reasons)
   Struct  in_addr
         Unsigned long  s_addr ;     //   that ' s a 32 _bit long, or 4 bytes
   } ;
```

این فیلد چهار بایتی (۳۲ بیتی) برای نگهداری آدرس IP بکار میرود میتوانستیم آن را به طور مستقیم به شکل unsigned long معرفی کنیم ولی به دلایل تاریخی به این صورت تعریف شده !

ماشینهای Big Eending و Little Ending

ساختار بسته های IP و TCP درقالب کلمات ۳۲ بیتی سازماندهی شوند ماشینهای متفاوتی که در دنیا وجود دارد کلمات دو بایتی و چهار بایتی را به روش یکسانی ذخیره نمی کنند . به عنوان مثال فرض کنید یک کلمه ۲ بایتی با دستور MOV (یا Load) به حافظه اصلی منتقل شود ماشینها به دوروش آن را درحافظه ذخیره میکنند

۱) ماشینهایی که بایت کم ارزش و سپس بایت پر ارزش را ذخیره میکنند، ماشینهای Little Ending نامیده میشوند

۲) ماشینهایی که بایت پرارزش وسپس بایت کم ارزش را ذخیره میکنند، ماشینهای Big Ending نامیده میشوند

پروتکل TCP/IP،استاندارد ماشینهای Big Ending رامبنا قرارداده لذا درتمام ماشینهای Little Ending قبل ازارسال بسته های IP بایدفیلدهای دوبایتی و چهاربایتی درون حافظه،به گونه ای تنظیم و مقداردهی شود تا درهنگام ارسال ابتدا بایت پرارزش ارسال شده و استانداردماشینهای B.E رعایت شود

- مثال:

وقتی روی ماشین از نوع LE دستور زیر اجرا شود:

```
Struct sockaddr _in   My_socket;
My_ socket.sin_ port =0xB459;
```

چون بایت کم ارزش ابتدا ذخیره میشود و بعد ازآن بایت پر ارزش قرار میگیرد پس نحوه قرارگیری آن در بسته TCP به صورت 59B4 تنظیم میشود که نادرست است به همین دلیل توابعی جهت رفع این مشکل معرفی شده است

htnos() _ “Host to Network Short”

تابع تبدیل کلمات دو بایتی به حالت BE

htonl() _ “Host to Network Long”

تابع تبدیل کلمات چهاربایتی به حالت BE

ntohs() _ “Network to Host Short”

تابع تبدیل کلمات دو بایتی از BE به حالت فعلی ماشین

ntohl() _ “Network to Host Long”

تابع تبدیل کلمات چهار بایتی از BE به حالت فعلی ماشین

مشکلات تنظیم آدرس IP درون فیلد آدرس

عموماً کاربران تمایل دارندآدرس IP را به صورت چهار عدد صحیح دهدهی که با . جدا میشوند وارد کنند و ماهیت آن یک رشته کاراکتری خواهد بود نه یک عدد صحیح چهار بایتی بدون علامت ! در حالی که در استراکچر sockaddr_in فیلد آدرس IP عددی است چهاربایتی که با یک عدد از نوع Long پرمیشود.

تابع inet_addr جهت تبدیل آدرسهای IP از رشته ای به عددی کاربرد دارد(یک رشته کاراکتری به شکل نقطه دار را گرفته و به یک عدد چهار بایتی با قالب BE تبدیل میکند).

```
ina.sin_addr.s_addr = inet_addr("10.12.110.57") ;
```

تابع inet_ntoa() عکس ، عمل تابع قبلی را انجام میدهد یعنی یک آدرس IP چهاربایتی در قالب BE را گرفته وآن را به یک رشته ی کاراکتری نقطه دار تبدیل میکند.

میتوان بجای استفاده از تابع inet_addr() از تابع خوش تعریف و مفیدتر inet_aton() استفاده کرد.

```
Char  *a1, *a2 ;

a1  =  inet_ntoa ( ina1 .sin_addr) ;      // this is 192.168.4.14
a2  =  inet_ntoa ( ina2 .sin_addr) ;      // this is 10.12.110.57
```

خروجی

```
address 1: 192.168.4.14
address 2 : 10.12.110.57
```

توابع مورد استفاده در برنامه سرویس دهنده (مبتنی بر سوکتهای استریم)

- تابع socket() فرم کلی آن:

```
Int  socket ( int domain , int  type  , int  protocol )
```

- Domain این پارامتر نشان دهنده خانواده سوکت است و در برنامه نویسی شبکه اینترنت با TCP/IP ، با مقدار ثابت PF_INET تنظیم میشود

- Type با این پارامتر نوع سوکت را اعلام میکند که میتواند نوع استریم یا از نوع دیتاگرام باشد socket_stream و یا stream_dgram باشد

- Protocol در این فیلد شماره شناسایی پروتکل مورد نظرتان را تنظیم میکنید که برای کاربردهای شبکه اینترنت همیشه مقدارآن صفراست

- اگر مقدار بازگشتی تابع socket() ، ۱- باشد عمل موفقیت آمیزنبوده کارباید متوقف شود ومتغیر سراسری errno() شماره ی خطای رخ داده را برمیگرداند

- تابع bind() معمولاً در برنامه سمت سرویس دهنده معنا دارد، شما ازاین طریق از سیستم عامل خواهش میکنید تمام بسته های TCPیا UDP با شماره پورت خاص را به سمت برنامه شما هدایت کند

```
Int bind (int sockfd, struct sockaddr *my_addr, int addrlen);
```

- Sockfd همان مشخصه سوکت است که با استفاده از تابع socket() باز کردید
- my_addr یک استراکچر که خانواده سوکت، آدرس پورت و آدرس IP ماشیم محلی را در خود نگه میدارد
- addrlen شامل طول استراکچر my_addr بر حسب بایت است

- تابع listen() این فقط در برنامه سرویس دهنده معنا می یابد ،(همان عمل گوش کردن و منتظر ماندن است) در یک عبارت ساده اعلام به سیستم عامل برای پذیرش تقاضاهای ارتباط TCP است سیستم عامل باید بداند که چند پروسه میتواند به طور همزمان ارتباط TCP به یک ادرس پورت داشته باشد و انها را در صف سرویس دهی قرار دهد.

```
Int listen ( int sockfd , int backlog );
```

- sockfd همان مشخصه سوکت است.
- backlog حداکثرتعداد ارتباط معلق و به صف شده ی منتظراست که در بسیاری از سیستمها مقدارbacklog به ۲۰ محدود شده.

(مانند توابع قبلی در صورت بروز خطا مقدار برگشتی این تابع ۱- خواهد بود و متغیر errno شماره خطای رخ داده را برمیگرداند)

تابع accept()

وقتی که تابع accept() اجرا میشئد برنامه شما از سیستم عامل می خواهد که از بین تقاضاهای به صف شده یکی را انتخاب کرده و آن را با مشخصات پروسه طرف مقابل ، تحویل برنامه تان بدهد پس تنها ۱ انتخاب وجود دارد بنابراین سیستم عامل یک مشخصه سوکت جدید ایجاد میکند ،
سوکت اول ← توسط تابع socket() ایجاد شده و سوکت دوم ← توسط تابع accept() به برنامه شما برگشته :

الف) سوکت اول شامل مشخصه تمام ارتباطات به صف شده ی منتظر است .
ب) سوکت دوم شامل مشخصه فقط یکی از اتصالات به صف شده ی معلق است.

```
int accept ( int  sockfd ,  Struct  sockaddr *addr ,  socklen_t  *addrlen);
```

addr استراکچری است که سیستم عامل پس از پذیرش یک ارتباط معلق آدرس پورت و آدرسIP طرف مقابل ارتباط را به برنامه شما برمیگرداند

addrlen طول استراکچر addr را بر حسب بایت مشخص میکند

- توابع send() و recv()

این دو تابع در برنامه سمت سرویس دهنده و مشتری یکسان بکارمیرود:

```
Int send ( int sockfd  , const void *msg ,  int len  ,  int flags)
Int recv (  int sockfd ,  void *buf ,  int len ,  unsigned int flags )
```

Sockfd مشخصه سوکتی است که از تابع accept() بدست آمده

msg این پارامتردرتابع send()، آدرس محلی درحافظه که داده های ارسالی از آنجا استخراج و داخل فیلد داده قرار گرفته و ارسال میشود را مشخص میکند .

len طول داده های ارسالی یا دریافتی بر حسب بایت را تعیین میکنید.

flag بدون توضیح فقط درآن صفربگذارید.

buf این پارامتر در تابع recv()، آدرس محلی درحافظه که داده های دریافتی درآنجا ست و به برنامه بازگردانده میشود

نکاتی مربوط به توابع send() و recv()

۱) این دو تابع فقط برای ارسال و دریافت روی سوکتهای نوع استریم کاربرد دارد.

۲) مقدار برگشتی این دو تابع در صورت خطا ۱- خواهد بود ولی در صورتی که مقدار برگشتی عددی مثبت باشد تعداد بایتهای ارسالی یا دریافتی را مشخص میکند.

۳) در هر مرحله سعی کنید حجم داده ها ارسالی توسط تابع send() تقریباً یک کیلو بایت باشد.

توابع close() و shutdown() :

- با استفاده از این دو تابع میتوانید بعد از اتمام نیاز ارتباط یا اتصال TCP را ببندید

```
Close (sockfd) ;
```

- Sockfd مشخصه سوکتی است که میخواهید آن را ببندید دقت کنید این سوکت همان مشخصه ای است که توسط تابع accept() برگشته اگر سوکتی باشد که توسط تابع socket() برگشته تمام ارتباطات معلق بسته میشود و امکان ارتباط به کل قطع میشود

سوکتی که توسط تابع close() بسته میشود دیگربرای ارسال و دریافت قابل استفاده نیست و سیستم عامل میتواند به جای آن تقاضای اتصال دیگری را قبول کند

- راه دیگر بستن یک سوکت تابع()shutdown است:

```
Int shutdown( int sockfd , int how);
```

- how روش بستن سوکت که یکی از مقادیر زیر است:

الف) مقدار صفر دریافت داده را غیرممکن میکند ولی سوکت برای ارسال داده، همچنان باز است.

ب) مقدار یک ارسال داده را غیرممکن میکند ولی سوکت برای دریافت داده، همچنان باز است.

ج) مقدار دو ارسال و دریافت را غیرممکن میکند سوکت کاملاً بسته میشود(مانند تابع ()close)

با تشکر از توجه شما

ادامه ارائه فصل هفتم

توسط محسن طاهری