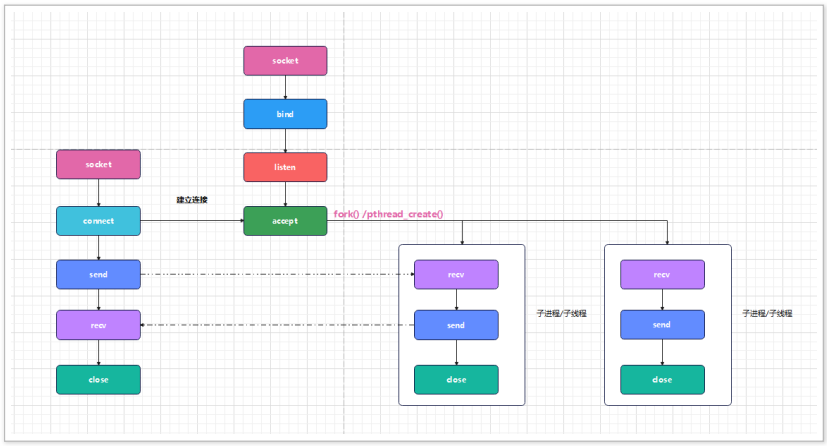
4.1 tcp 并发服务器 - 多进程_物联网 / 嵌入式工程师 - 慕课网

> 慕课网慕课教程 4.1 tcp 并发服务器 – 多进程涵盖海量编程基础技术教程，以图文图表的形式，把晦涩难懂的编程专业用语，以通俗易懂的方式呈现给用户。

- 并发服务器的设计思想:
  - 每个客户端与服务器建立连接之后，服务器都会创建一个子进程 / 子线程来进行相应的数据处理
- 并发服务器最大的优势还是在于提高了服务器处理请求的效率
- 并发服务器框架如下:
- 

- tcp 并发服务器基于多进程实现如下:
  - step 1 : 当 服务器与客户端建立连接后，则会创建进程
  - ```
    for(;;){

            bzero(&cli_addr,sizeof(struct sockaddr));
            cfd = accept(sfd,(struct sockaddr *)&cli_addr,&len);
            if (cfd == -1){
                perror("[ERROR] accept(): ");
                exit(EXIT_FAILURE);
            }

            printf("ip : %s,port : %d\n",inet_ntoa(cli_addr.sin_addr),ntohs(cli_addr.s

            cpid = fork();
            if (cpid == -1){
                perror("[ERROR] fork(): ");
                close(cfd);
            }else if (cpid == 0){

            }
        }
    ```
  - step 2 : 实现客户端数据处理函数, 并在子进程中执行
    - 客户端数据处理函数实现
    - ```
      void do_client(int cfd)
      {
          char buffer[1024] = {0};
          ssize_t sbytes,rbytes;

          memset(buffer,0,sizeof(buffer));
          rbytes = recv(cfd,buffer,sizeof(buffer),0);
          if (rbytes == -1){
      ```

```
            perror("recv(): ");
            close(cfd);
            exit(EXIT_FAILURE);
        }else if (rbytes == 0){
            printf("The client is offline.\n");
            close(cfd);
            exit(EXIT_FAILURE);

        }else if (rbytes > 0){
            sbytes = send(cfd,buffer,sizeof(buffer),0);
            if (sbytes == -1){
                perror("[ERROR] send(): ");
                close(cfd);
                exit(EXIT_FAILURE);
            }
        }

        close(cfd);
        exit(EXIT_SUCCESS);
    }
```

- 让客户端处理函数在子进程中执行
- ```
  for(;;){

      bzero(&cli_addr,sizeof(struct sockaddr));
      cfd = accept(sfd,(struct sockaddr *)&cli_addr,&len);
      if (cfd == -1){
          perror("[ERROR] accept(): ");
          exit(EXIT_FAILURE);
      }

      printf("ip : %s,port : %d\n",inet_ntoa(cli_addr.sin_addr),ntohs(c
      cpid = fork();
      if (cpid == -1){
          perror("[ERROR] fork(): ");
          close(cfd);
      }else if (cpid == 0){
          do_client(cfd);
      }
  }
  ```

- step 3：通过信号来处理僵死进程
- ```
  void do_sigchld_handler(int sig)
  {
      wait(NULL);
  }

  int main()
  {
      __sighandler_t retsig;

      retsig = signal(SIGCHLD,do_sigchld_handler);
      if (retsig == SIG_ERR){
          perror("[ERROR] signal(): ");
          exit(EXIT_FAILURE);
      }


      return 0;
  }
  ```

全文完