



**Facultad de Ingeniería en
Electricidad y Computación**

Programación de Sistemas

CCPG1051

Federico Domínguez, PhD.

Unidad 6 – Sesión 5: Creación de un servicio multi-clientes

Agenda

Daemons

Múltiples clientes con hilos

Un *daemon* es un programa que corre en segundo plano en lugar de interactuar directamente con el usuario.

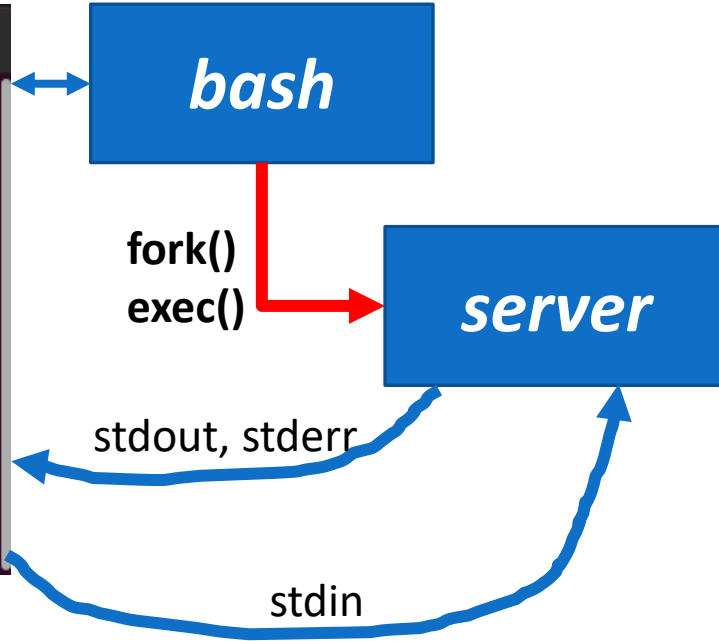
En sistemas UNIX/LINUX un *daemon* típicamente ofrece un servicio al sistema o implementa un proceso de mantenimiento.

Por convención, el nombre de programas que funcionan como *daemon* terminan con la letra “d”: *syslogd*, *httpd*, *mysqld*

Procesos conocidos como servicios funcionan usualmente en modo *daemon*.

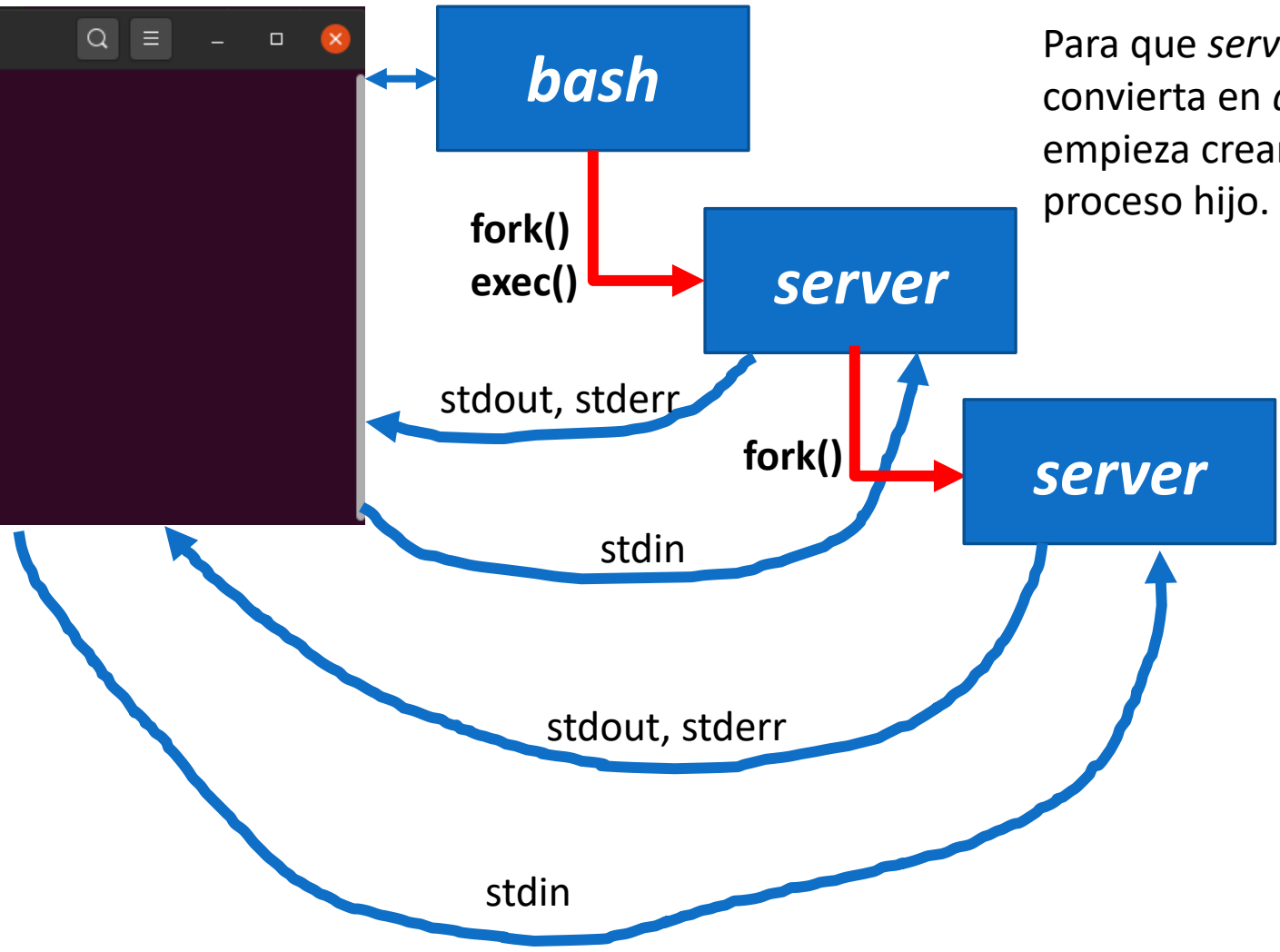


```
federico@federico-ubuntu20: ~  
federico@federico-ubuntu20:~$ ./server 8080  
server escuchando en puerto 8080...  
_
```



Al ejecutar un programa, *bash* crea un proceso nuevo el cual es hijo de *bash*. En este ejemplo, el programa *server* se ejecuta empezando como un proceso interactivo ligado a la consola.

```
federico@federico-ubuntu20: ~  
federico@federico-ubuntu20:~$ ./server 8080  
server escuchando en puerto 8080...
```



Para que *server* se convierta en *daemon*, empieza creando un proceso hijo.

```
federico@federico-ubuntu20: ~  
federico@federico-ubuntu20:~$ ./server 8080  
server escuchando en puerto 8080...  
Entrando modo daemon...  
federico@federico-ubuntu20:~$
```

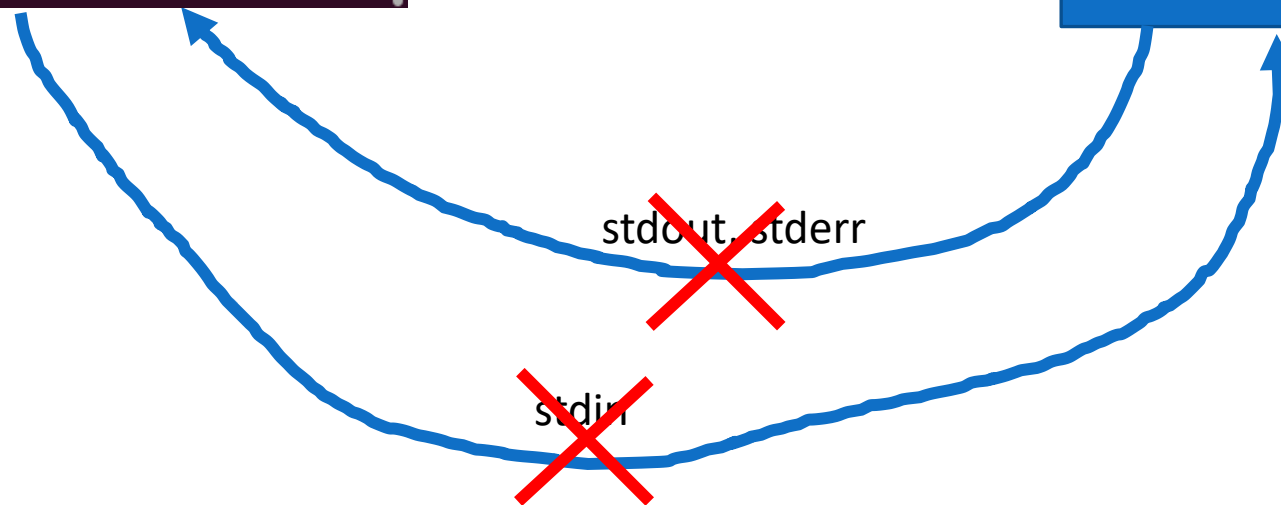
bash

server

Inmediatamente el
proceso padre termina y
el proceso hijo cierra
stdout, *stderr*, *stdin*.

exit(0)

server



```
federico@federico-ubuntu20: ~  
federico@federico-ubuntu20:~$ ./server 8080  
server escuchando en puerto 8080...  
Entrando modo daemon...  
federico@federico-ubuntu20:~$
```

bash

Ahora, *server* es un *daemon* corriendo en segundo plano, independiente de la consola.

server

```
federico@federico-ubuntu20: ~  
federico@federico-ubuntu20:~$ ./server 8080  
server escuchando en puerto 8080...  
Entrando modo daemon...  
federico@federico-ubuntu20:~$
```

bash

Es buena práctica usar
ahora *syslog* en el nuevo
daemon como bitácora.

server

syslog

Ejemplo de función que implementa modo *daemon*...

```
void daemonize(char *nombre_programa)
{
    printf("Entrando modo daemon...\n");

    int fd0, fd1, fd2;
    pid_t pid;

    if ((pid = fork()) < 0){
        fprintf(stderr, "No es posible hacer un fork, error %s\n", strerror(errno));
        exit(1);
    }else if (pid != 0)
        exit(0);

    setsid();

    //Cerrar solamente stdout, stdin y stderr
    close(0);
    close(1);
    close(2);

    //se deben abrir otra vez, porque se podría abrir un archivo al stdout, stdin o stderr
    //y los printf van a enviarían su output a ese archivo
    fd0 = open("/dev/null", O_RDWR);
    fd1 = dup(fd0);
    fd2 = dup(fd0);

    //Abrir un log para este daemon en el sistema syslog
    openlog(nombre_programa, LOG_CONS, LOG_DAEMON);
    if (fd0 != 0 || fd1 != 1 || fd2 != 2) {
        syslog(LOG_ERR, "unexpected file descriptors %d %d %d",
            fd0, fd1, fd2);
        exit(1);
    }
}
```

Existe además una versión de esta función en la librería `unistd.h`

daemon(3) — Linux manual page

[NAME](#) | [SYNOPSIS](#) | [DESCRIPTION](#) | [RETURN VALUE](#) | [ATTRIBUTES](#) | [CONFORMING TO](#) | [NOTES](#) | [BUGS](#) | [SEE ALSO](#) | [COLOPHON](#)

DAEMON(3)

Linux Programmer's Manual

DAEMON(3)

NAME [top](#)

daemon - run in the background

SYNOPSIS [top](#)

```
#include <unistd.h>
```

```
int daemon(int nochdir, int noclose);
```

Feature Test Macro Requirements for glibc (see [feature_test_macros\(7\)](#)):

```
daemon():
```

```
    Since glibc 2.21:
```

```
    _DEFAULT_SOURCE
```

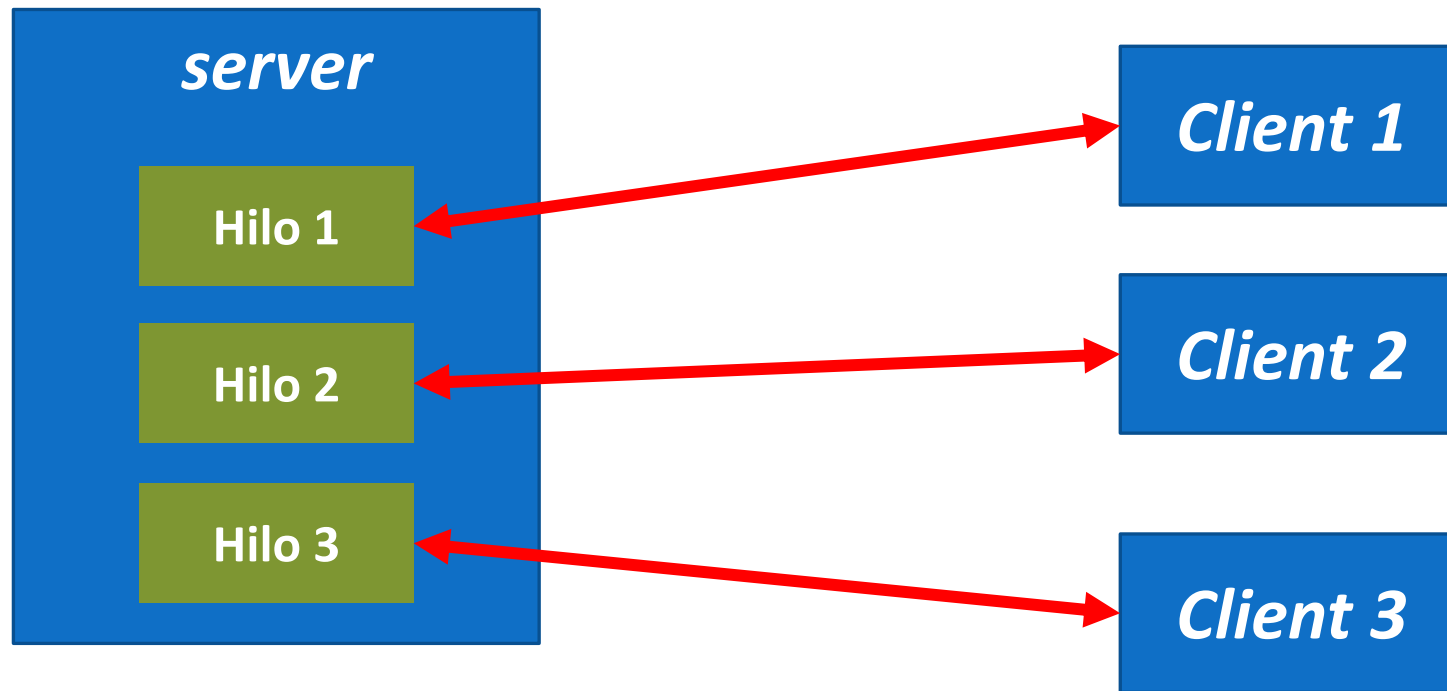
```
    In glibc 2.19 and 2.20:
```

```
    _DEFAULT_SOURCE || (_XOPEN_SOURCE && _XOPEN_SOURCE < 500)
```

```
    Up to and including glibc 2.19:
```

```
    _BSD_SOURCE || (_XOPEN_SOURCE && _XOPEN_SOURCE < 500)
```

Un servicio concurrente (múltiples clientes) con hilos...



Un servicio concurrente (múltiples clientes) con hilos...

```
6  int main(int argc, char **argv)
7  {
8      int listenfd, *connfdp, port;
9      socklen_t clientlen=sizeof(struct sockaddr_in);
10     struct sockaddr_in clientaddr;
11     pthread_t tid;
12
13     if (argc != 2) {
14         fprintf(stderr, "usage: %s <port>\n", argv[0]);
15         exit(0);
16     }
17     port = atoi(argv[1]);
18
19     listenfd = Open_listenfd(port);
20     while (1) {
21         connfdp = Malloc(sizeof(int));
22         *connfdp = Accept(listenfd, (SA *) &clientaddr, &clientlen);
23         Pthread_create(&tid, NULL, thread, connfdp);
24     }
25 }
```

```
26
27  /* Thread routine */
28  void *thread(void *vargp)
29  {
30      int connfd = *((int *)vargp);
31      Pthread_detach(pthread_self());
32      Free(vargp);
33      echo(connfd);
34      Close(connfd);
35      return NULL;
36  }
```

Referencias

Libro guía *Computer Systems: A programmers perspective*. Sección 12.3.8

<https://www.man7.org/linux/man-pages/man3/daemon.3.html>