Ciências ULisboa | Informática

Sockets com Múltiplos Clientes

Pedro Ferreira, Vinicius Cogo

Tópicos

1. **Suportar Múltiplos Clientes**
2. **Servidores Concorrentes**
3. **Multiplexagem de I/O em Python**

1. Suportar Múltiplos Clientes

Como suportar **múltiplos clientes** num servidor?

- Servidor **não concorrente**
  - Aceita múltiplos clientes,
    mas trata um pedido de cada vez
    (inaceitável para servidores reais com muita procura)

- Servidor **concorrente**
  - Trata pedidos de múltiplos clientes em simultâneo

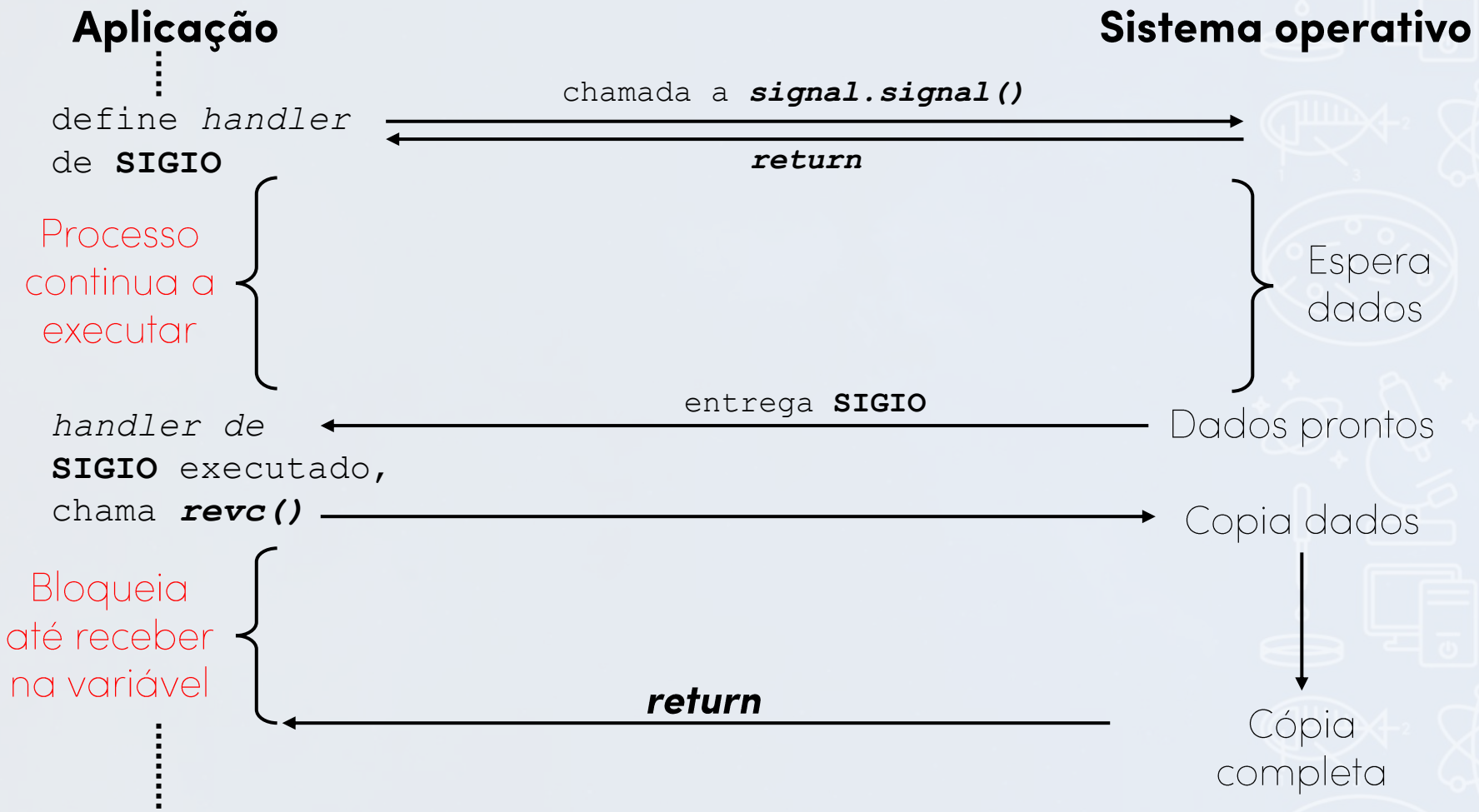
2. Servidores Concorrentes

Como implementar **servidores concorrentes**?

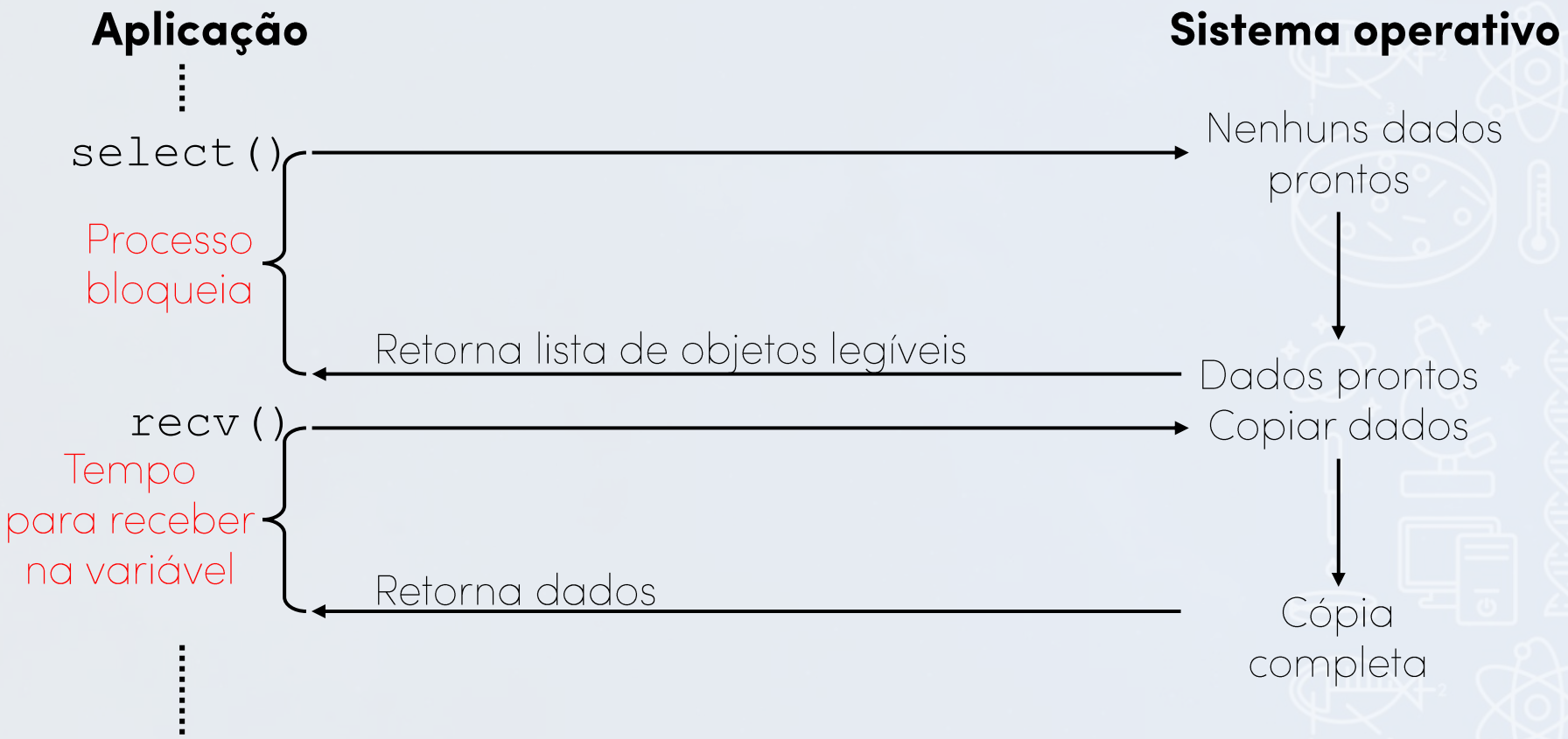
- Múltiplos **fios de execução**
  - ***Threads***
  - **Processos** (*fork*)

- **Multiplexagem** de *Input/Output* (I/O)
  - I/O não bloqueante **com sinais**
  - I/O não bloqueante **com *select***, ***poll*** e similares

Multiplexagem de I/O com Sinais



Multiplexagem de I/O com Select



3. Multiplexagem de I/O em Python

- Módulo select
  - Dá acesso a um conjunto de funções do sistema que permitem **monitorizar múltiplos descritores de ficheiros**, até que pelo menos um deles esteja pronto para alguma operação de I/O
  - As funções do módulo têm portabilidade variável
    - As mais portáveis são `select` e `poll`
    - Windows só suporta `select`
  - Vamos usar `select`

Ciências ULisboa

Utilização da chamada **select**

```
import select

R, W, X = select.select(rlist, wlist, xlist, timeout)
```

rlist: lista de objetos monitorizados para leitura
wlist: lista de objetos monitorizados para escrita
xlist: lista de objetos monitorizados para exceções
timeout: se for 0, faz uma monitorização e sai; se for omitido, a função bloqueia até que um dos objetos esteja pronto; valor de virgula flutuante > 0, é um timeout

R, **W** e **X**: Listas com os objetos prontos. São subconjuntos de rlist, wlist e xlist.

Multiplexagem de I/O em Python

Cliente interativo

(Estabelecimento da ligação omitido. Similar às aulas anteriores)

```
addr, port = sock.getsockname()
print('Ligado pelo endereço local %s:%d' % (addr, port))

msg = 'Go'
while msg:
  msg = input('Mensagem: ')

  if msg == 'EXIT':
    msg = ''
  else:
    sock.sendall(msg)
    resposta = sock.recv(1024)
    print('Resposta: %s' % resposta)

sock.close()
```

Multiplexagem de I/O em Python

Servidor com multiplexagem

(omitido da criação da socket de escuta até à chamada a listen(). Similar às aulas anteriores.)

```
import select as sel

SocketList = [ListenSocket]
while True:
    R, W, X = sel.select(SocketList, [], []) # Espera sockets com
    for sckt in R:                           # dados para leitura
        if sckt is ListenSocket:     # Se for a socket de escuta
            conn_sock, (addr, port) = ListenSocket.accept()
            print('Novo cliente ligado desde %s:%d' % (addr, port))
            SocketList.append(conn_sock)   # Adiciona ligação à lista
        else:                            # Se for a socket de um cliente
            msg = sckt.recv(1024)
            if msg:                          # Se recebeu dados
                sckt.sendall(msg[::-1])      # responde
            else:                            # Se não recebeu dados
              sckt.close()                       # cliente fechou ligação
              SocketList.remove(sckt)
              print('Cliente fechou ligação')
```

Bibliografia

- Brandon Rhodes and John Goerzen. Foundations of Python Network Programing, second edition, Apress.

- Python online documentation:select —Waiting for I/O completion. (https://docs.python.org/3/library/select.html)