Oscar García Lorenzo

Grado en Robótica

Universidade de Santiago de Compostela

Redes e comunicacións, 3º Curso Gri

citius.usc.es







Índice

Procesos e Fíos

2 Servidor HTTP



Índice

Procesos e Fíos

2 Servidor HTTP



Procesos

- Unidade básica de execución
- O SO empeza cun proceso:
 - Ese proceso dividese e convirtese en diversos programas
 - Unha terminal é un proceso
 - Ao executar algo duplicase e un deles convirtese no executado
 - ▶ Se se termina/mata ao proceso pai termínanse/matan os fillos
- A memoria dos procesos é completamente independente
 - As variabeis son independentes
 - Zonas dinámicas de memoria separadas
- Os arquivos/sockets non independentes
 - Os arquivos/sockets se pechan cando non hai ningún proceso con eles abertos
- Identificanse cun PID (linux)



Fíos

- Unidade minima de execución (procesos lixeiros)
- Empezase cun proceso:
 - Ese proceso lanza diversos fíos
 - O fío inicial do proceso é igual ao resto
 - ▶ Se se termina/mata fío inicial o resto dos fíos continúan
- A memoria dos fíos é compartida (case toda)
 - As variabeis son compartidas, a non ser que se declaren localmente
 - Zonas dinámicas de memoria compartidas
- Os arquivos/sockets compartidos
- Identificanse cun TID, comparten PID (linux)



Carreiras críticas

- Non se pode asegurar cando se vai executar un proceso/fío
- O orde de acceso a recursos compartidos non se coñece
- Hai que sincronizar
- Nos non o faremos:
 - ▶ TCP pensado para elo
 - Socket de servidor e de conexión
 - O proceso fillo/fio usa el so un socket de conexión
 - O pai o pecha
 - Non se comparten
 - Coidado ao escribir en ficheiros, ler non hai problema (punteiros de lectura diferentes)



Procesos e Fíos Servidor HTT

Procesos Python

- Librarías que permiten un control fino
- Usaremos á máis parecida a C, ao que fai realmente o SO
- Libraría OS (en linux, para Windows funciona diferente)
- os.fork()
 - Duplica o proceso
 - Devolve un (0) ao proceso fillo
 - Devolve o PID ao proceso pai
- os.wait()
 - Espera a que acabe 1 proceso fillo (calquera)
 - ▶ Devolve o par (PID, status)
- os.waitpid(pid, options)
 - Espera por un PID en concreto
 - As opcións poden ser varias
 - os.WNOHANG:
 - A función retorna immediatamente se non hai status para ese proceso, non espera



Procesos e Fíos Servidor HTTP

Procesos Sinais

- As sinais sincronizan cousas do SO (periféricos, procesos, comunicacións ...)
- Poden lanzarse e recollerse
- Asíncronas
- Exemplo: time.sleep() programa unha sinal SIGALRM en tantos segunda e pausa o proceso/fío, cando pasa ese tempo o SO envía unha sinal SIGALRM ao proceso, que a recolle e procesa
- Cando un proceso fillo termina manda unha mensaxe SIGCHLD
- Esa sinal pode recollerse asíncronamente (no momento que chega)
- Libraría signal
- signal.signal(signalnum, handler)
 - Di que facer (función handler) cando se recibe a sinal signalnum
 - Pausa a execución normal do proceso, procesa a sinal e logo continúa



Procesos e Fíos Servidor HTTP

Fíos Python

- Hai librarías que permiten un control fino e mellor
- Usaremos á máis parecida a C, pero por como xestiona Python a memoria non funciona igual
- Libraría Threading (en linux, para Windows funciona diferente)
- threading.Thread(group=None, target=None, name=None, args=(), kwargs=, *, daemon=None)
 - Crea un fío
 - ▶ target pode ser unha función
- threading.run()
 - Inicia e execución do fío
- threading.join(timeout=None)
 - Espera a que acabe ese fío
- threading.is_alive()
 - ▶ Comproba se o fío está en execcución
- threading.get_native_id()
 - Devolve o TID, pero so para Python 3.8



Exemplos

- exemplo_fork_1.py
 - Lanza 3 procesos fillos
 - O pai espera a que terminen, garda os PID nunha lista e recolle as sinais dos fillos
 - Modifican unha variable para ver que pasa
 - ▶ Ten comentada unha versión con wait () para ver que ocorre
- exemplo_fork_2.py
 - Igual ao anterior, pero con variable global, igual que o seguinte para comparar
- exemplo_threads.py
 - Lanza 3 fíos extra
 - O 1º fío espera a que terminen, garda os ID nunha lista e fai join ()
 - Modifican unha variable para ver que pasa
 - ▶ Ten comentada unha versión con is_alive() para ver que ocorre



Procesos e Fíos Servidor HTTI

Exemplos

- Para usar sockets acordarse de facer o socket herdable con socket.set inheritable(True) antes do fork()
- Por defecto os sockets que se crean non se poden herdar polo fillos
 - ▶ Isto é cousa de Python 3.4, que o fai para todos os descriptores de ficheiros, no resto de linguaxes non soe ser así



Índice

1 Procesos e Fíos

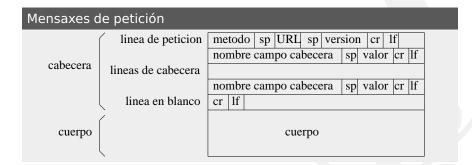
2 Servidor HTTP



- Un servidor HTTP funciona con TCP
- As mensaxes HTTP son so texto
- En HTTP 1.0 (conexión non permanente):
 - Realizase a conexión, crease un socket de conexión
 - Leese do socket de conexión unha mensaxe (HTTP)
 - Procesase a mensaxe HTTP
 - Crease unha mensaxe de resposta
 - Envíase a mensaxe de reposta
 - Péchase a conexión (socket de conexión)



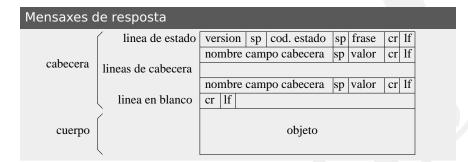
Mensaxes HTTP





cesos e Fíos Servidor HTTP

Mensaxes HTTP





- Servidor simple:
 - response = 'HTTP/1.0 200 OK\n\nHello World'
- Servidor máis complexo:
 - P.e. usar mensaxe = str(mensaxeCompleta).split('
 ')
 - ▶ Se mensaxe[0] == 'GET'
 - Se mensaxe[1]=='/' → mensaxe[1]='index.html'
 - Abrir o ficheiro chamado mensaxe [1] e le-lo
 - data = in_file.read() o garda como ASCII
 - A mensaxe de resposta ten que levar:
 - Content-Type (os navegadores son listos, funcionará igual)
 - Content-Length tamaño de data
 - data ao final
 - Envíase esa resposta

