ΑΕΙ Πειραιά Τ.Τ.

Τμήμα Μηχανικών Ηλεκτρονικών Υπολογιστικών Συστημάτων

Εργαστήριο Λειτουργικών Συστημάτων

Εργαστηριακή άσκηση
Υπηρεσίες - Δαίμονες

Επιμέλεια:
Στυλιανός Βουτσινάς

BSc,MSc,PhD(Cand.)

Υπεύθυνος Εργαστηρίου:

Δρ. Διλιντάς Γεωργιος (Καθηγητής)

Αναλυτική Περιγραφή

Ένας δαίμονας (daemon) είναι μια διεργασία που εκτελείται στο παρασκήνιο έχοντας αποσυνδεθεί από την τυπική είσοδο/έξοδο και δεν λειτουργεί από τον άμεσο έλεγχο ενός χρήστη. Περιμένει να ενεργοποιηθεί από την εμφάνιση ενός συγκεκριμένου γεγονότος ή συνθηκών.

Λειτουργικά συστήματα συμβατά με το UNIX χρησιμοποιούν πολυάριθμους δαίμονες, κυρίως για να εξυπηρετούν αιτήσεις για υπηρεσίες από άλλους υπολογιστές σε ένα δίκτυο, αλλά και για να ανταποκρίνονται σε άλλες διεργασίες και σε δραστηριότητες υλικού. Παραδείγματα ενεργειών ή συνθηκών που μπορούν να ενεργοποιήσουν τους δαίμονες σε δραστηριότητα είναι μια συγκεκριμένη ώρα ή ημερομηνία, η μετάβαση ενός καθορισμένου χρονικού διαστήματος, η προσπέλαση ενός αρχείου σε έναν συγκεκριμένο κατάλογο, η λήψη ενός ηλεκτρονικού ταχυδρομείου. Δεν είναι απαραίτητο ο δράστης της ενέργειας ή της κατάστασης να γνωρίζει ότι ένας δαίμονας ακούει, παρόλο που τα προγράμματα συχνά εκτελούν μια ενέργεια μόνο επειδή γνωρίζουν ότι θα προκαλέσουν σιωπηρά έναν δαίμονα.

Οι δαίμονες συνηθίζεται το όνομα τους να τελειώνει με τον χαρακτήρα d (πχ httpd) και αναγνωρίζονται από το σύστημα ως κάθε διεργασία της οποίας η γονική διεργασία έχει ένα PID ίσο με 1, το οποίο αντιπροσωπεύει πάντα τη διεργασία init. Η init είναι πάντα η πρώτη διεργασία που εκκινεί και παραμένει στο σύστημα μέχρι να απενεργοποιηθεί ο υπολογιστής. Η init υιοθετεί οποιαδήποτε διεργασία της οποίας η γονική διεργασία πεθαίνει (δηλ. τερματίζεται) χωρίς να περιμένει τον κωδικό τερματισμού της θυγατρικής διεργασίας. Έτσι, η κοινή μέθοδος για την εκκίνηση ενός δαίμονα περιλαμβάνει τη δημιουργία θυγατρικής διεργασίας παράλληλα με τον τερματισμό της γονικής της διεργασίας. Στη συνέχεια η θυγατρική διεργασία ξεκινά την κανονική λειτουργία της.

Τέτοιες διεργασίες ελέγχονται από scripts τα οποία δίνουν την δυνατότητα ελέγχου στο χρήστη (start|stop|restart|status)

***Τεκμηρίωση* Μεταβλητών**

DAEMON_PATH="$HOME/serviceName"

Μονοπάτι που βρίσκεται αποθηκευμένο το εκτελέσιμο της υπηρεσίας
Ορισμός στη γραμμή 6 του αρχείου servicectl.sh.

DAEMON=serv

Όνομα εκτελέσιμου
Ορισμός στη γραμμή 9 του αρχείου servicectl.sh.

DAEMONOPTS=""

Πιθανές επιλογές υπηρεσίας
Ορισμός στη γραμμή 12 του αρχείου servicectl.sh.

PIDFILE=$HOME/serviceName/daemon.pid

Μονοπάτι της τοποθεσίας του κανονικού αρχείου που θα φέρει το PID της υπηρεσίας
Ορισμός στη γραμμή 15 του αρχείου servicectl.sh.

Τεκμηρίωση Συναρτήσεων

start ()

Εκκίνηση της υπηρεσίας. Ορισμός στη γραμμή 19 του αρχείου servicectl.sh.

```
    {
20      printf "%-50s" "Starting $DAEMON..." #εκτύπωση μηνύματος 50 χαρακτήρες με
αριστερή στοίχιση
21      cd $DAEMON_PATH
22      ./$DAEMON $DAEMONOPTS
23      PID=`cat $PIDFILE`
24          if [ -z $PID ]; then
25              printf "%s\n" "[Failed]"
26          else
27              printf "%s\n" "[Ok]"
28              printf "%s\n" "[$PID]"
29          fi
30  }
```

stop ()

Τερματισμός της υπηρεσίας. Ορισμός στη γραμμή 34 του αρχείου servicectl.sh.

```
34  {
35          printf "%-50s" "Stopping $DAEMON"
36              PID=`cat $PIDFILE`
37              cd $DAEMON_PATH
38          if [ -f $PIDFILE ]; then
39              kill $PID
40              printf "%s\n" "[OK]"
41              rm -f $PIDFILE
42          else
43              printf "%s\n" "pidfile not found"
44          fi
45  }
```

status ()

Εκτυπώνει την κατάσταση λειτουργίας της υπηρεσίας. Ορισμός στη γραμμή 49 του αρχείου servicectl.sh.

```
49  {
50          printf "%-50s" "Checking $DAEMON..."
51          if [ -f $PIDFILE ]; then
52              PID=`cat $PIDFILE`
53              if [ -z "`ps -axf | grep ${PID} | grep -v grep`" ]
54              then
55                  printf "%s\n" "Process dead but pidfile exists"
56              else
57                  echo "Running"
58              fi
59          else
60              printf "%s\n" "Service not running"
61          fi
62  }
```

Κώδικας άσκησης servicectl.sh.

```
 1 #!/bin/bash
 2 ## @author Στυλιανός Βουτσινάς
 3 ## @file servicectl.sh
 4
 5 ##@brief Μονοπάτι που βρίσκεται αποθηκευμένο το εκτελέσιμο της υπηρεσίας
 6 DAEMON_PATH="$HOME/serviceName"
 7
 8 ##@brief Όνομα εκτελέσιμου
 9 DAEMON=serv
10
11 ##@brief Επιλογές
12 DAEMONOPTS=""
13
14 ##@brief PIDFILE
15 PIDFILE=$HOME/serviceName/daemon.pid
16
17 ##@fn start()
18 ##@brief Εκκίνηση της υπηρεσίας
19 start() {
20     printf "%-50s" "Starting $DAEMON..." #εκτυπωση μηνυματος 50 χαρακτήρες με
αριστερή στοίχηση
21     cd $DAEMON_PATH
22     ./$DAEMON $DAEMONOPTS
23     PID=`cat $PIDFILE`
24         if [ -z $PID ]; then
25             printf "%s\n" "[Failed]"
26         else
27             printf "%s\n" "[Ok]"
28             printf "%s\n" "[$PID]"
29         fi
30 }
31
32 ##@fn stop()
33 ##@brief Τερματισμός της υπηρεσίας
34 stop() {
35         printf "%-50s" "Stopping $DAEMON"
36             PID=`cat $PIDFILE`
37             cd $DAEMON_PATH
38         if [ -f $PIDFILE ]; then
39             kill $PID
40             printf "%s\n" "[OK]"
41             rm -f $PIDFILE
42         else
43             printf "%s\n" "pidfile not found"
44         fi
45 }
46
47 ##@fn status()
48 ##@brief Κατάσταση της υπηρεσίας
49 status(){
50         printf "%-50s" "Checking $DAEMON..."
51         if [ -f $PIDFILE ]; then
52             PID=`cat $PIDFILE`
53             if [ -z "`ps axf | grep ${PID} | grep -v grep`" ]
54             then
55                 printf "%s\n" "Process dead but pidfile exists"
56             else
57                 echo "Running"
58             fi
59         else
60             printf "%s\n" "Service not running"
61         fi
62 }
63
64 ##@brief Κλήση συναρτήσεων
65 case "$1" in
66   start)
67         start
68         ;;
69   stop)
70         stop
71         ;;
```

```
72   status)
73         status
74         ;;
75   restart|reload)
76         stop
77         start
78         ;;
79   *)
80         echo $"Usage: $0 {start|stop|restart|reload|status}"
81         exit 1
82 esac
83 exit 0
```

Κώδικας υπηρεσίας serv.c

```
 1 #include <stdio.h>
 2 #include <signal.h>
 3 #include <syslog.h>
 4 #include <errno.h>
 5 #include <stdlib.h>
 6 #include <fcntl.h>
 7 #include <unistd.h>
 8 #include <string.h>
 9 #include <sys/types.h>
10 #include <sys/stat.h>
11
12 #define DAEMON_NAME "simpledaemon"
13
14
15 void daemonShutdown();
16 void signal_handler(int sig);
17 void daemonize(char *rundir, char *pidfile);
18
19
20 int pidFilehandle;
21
22
23 void signal_handler(int sig)
24 {
25   switch(sig)
26   {
27     case SIGHUP:
28     syslog(LOG_WARNING, "Received SIGHUP signal.");
29     break;
30     case SIGINT:
31     case SIGTERM:
32     syslog(LOG_INFO, "Daemon exiting");
33     daemonShutdown();
34     exit(EXIT_SUCCESS);
35     break;
36     default:
37     syslog(LOG_WARNING, "Unhandled signal %s", strsignal(sig));
38     break;
39   }
40 }
41
42 void daemonShutdown()
43 {
44   close(pidFilehandle);
45 }
46
47 void daemonize(char *rundir, char *pidfile)
48 {
49   int pid, sid, i;
50   char str[10];
51   struct sigaction newSigAction;
52   sigset_t newSigSet;
53
54         /* Check if parent process id is set */
55   if (getppid() == 1)
56   {
57           /* PPID exists, therefore we are already a daemon */
58     return;
59   }
60
61         /* Set signal mask - signals we want to block */
62   sigemptyset(&newSigSet);
63         sigaddset(&newSigSet, SIGCHLD);  /* ignore child - i.e. we don't need to
wait for it */
64         sigaddset(&newSigSet, SIGTSTP);  /* ignore Tty stop signals */
65         sigaddset(&newSigSet, SIGTTOU);  /* ignore Tty background writes */
66         sigaddset(&newSigSet, SIGTTIN);  /* ignore Tty background reads */
67         sigprocmask(SIG_BLOCK, &newSigSet, NULL);   /* Block the above specified
signals */
68
69         /* Set up a signal handler */
70   newSigAction.sa_handler = signal_handler;
71   sigemptyset(&newSigAction.sa_mask);
```

```
   72   newSigAction.sa_flags = 0;
   73
   74              /* Signals to handle */
   75              sigaction(SIGHUP, &newSigAction, NULL);     /* catch hangup signal
*/
   76              sigaction(SIGTERM, &newSigAction, NULL);    /* catch term signal */
   77              sigaction(SIGINT, &newSigAction, NULL);     /* catch interrupt
signal */
   78
   79
   80         /* Fork*/
   81   pid = fork();
   82
   83   if (pid < 0)
   84   {
   85              /* Could not fork */
   86     exit(EXIT_FAILURE);
   87   }
   88
   89   if (pid > 0)
   90   {
   91              /* Child created ok, so exit parent process */
   92     printf("Child process created: %d\n", pid);
   93     exit(EXIT_SUCCESS);
   94   }
   95
   96         /* Child continues */
   97
   98         umask(027); /* Set file permissions 750 */
   99
  100         /* Get a new process group */
  101   sid = setsid();
  102
  103   if (sid < 0)
  104   {
  105     exit(EXIT_FAILURE);
  106   }
  107
  108         /* close all descriptors */
  109   for (i = getdtablesize(); i >= 0; --i)
  110   {
  111     close(i);
  112   }
  113
  114         /* Route I/O connections */
  115
  116         /* Open STDIN */
  117   i = open("/dev/null", O_RDWR);
  118
  119         /* STDOUT */
  120   dup(i);
  121
  122         /* STDERR */
  123   dup(i);
  124
  125         chdir(rundir); /* change running directory */
  126
  127         /* Ensure only one copy */
  128   pidFilehandle = open(pidfile, O_RDWR|O_CREAT, 0600);
  129
  130   if (pidFilehandle == -1 )
  131   {
  132              /* Couldn't open lock file */
  133     syslog(LOG_INFO, "Could not open PID lock file %s, exiting", pidfile);
  134     exit(EXIT_FAILURE);
  135   }
  136
  137         /* Try to lock file */
  138   if (lockf(pidFilehandle,F_TLOCK,0) == -1)
  139   {
  140              /* Couldn\'t get lock on lock file */
  141     syslog(LOG_INFO, "Could not lock PID lock file %s, exiting", pidfile);
  142     exit(EXIT_FAILURE);
  143   }
  144
  145
  146         /* Get and format PID */
```

```
147   sprintf(str,"%d\n",getpid());
148
149         /* write pid to lockfile */
150   write(pidFilehandle, str, strlen(str));
151 }
152
153 int main()
154 {
155         /* Debug logging
156         setlogmask(LOG_UPTO(LOG_DEBUG));
157         openlog(DAEMON_NAME, LOG_CONS, LOG_USER);
158         */
159
160         /* Logging */
161   setlogmask(LOG_UPTO(LOG_INFO));
162   openlog(DAEMON_NAME, LOG_CONS | LOG_PERROR, LOG_USER);
163
164   syslog(LOG_INFO, "Daemon starting up");
165
166         /* Deamonize */
167   daemonize(".", "./daemon.pid");
168
169   syslog(LOG_INFO, "Daemon running");
170
171   while (1)
172   {
173     syslog(LOG_INFO, "daemon says hello");
174
175     sleep(1);
176   }
177 }
```

Για την μεταγλώττιση του προγράμματος πληκτρολογούμε:
gcc serv.c -o serv