```
 1: // $Id: childwait.cpp,v 1.2 2014-05-23 12:26:39-07 - - $
 2:
 3: //
 4: // A zombie process is one that has exited but for which the
 5: // parent process has not waited for it.  This process creates
 6: // up to ten child processes and then waits for them.
 7: //
 8: // Usage: childwait children pdelay loops cdelays
 9: // children - number of children to create
10: // pdelay   - parent delays (seconds) before creating each child
11: // loops    - number of times each child is to print a message
12: // cdelays  - delay (seconds) before each child's message
13: //
14:
15: #include <cerrno>
16: #include <cstring>
17: #include <iomanip>
18: #include <iostream>
19: #include <sstream>
20: #include <stdexcept>
21: #include <string>
22: #include <typeinfo>
23: #include <vector>
24: using namespace std;
25:
26: #include <libgen.h>
27: #include <sys/time.h>
28: #include <sys/types.h>
29: #include <sys/wait.h>
30: #include <time.h>
31: #include <unistd.h>
32:
33: #include "../simple-sockets/logstream.h"
34: logstream outlog (cout);
35: logstream errlog (cerr);
36: string progname;
37: timeval start_time;
38:
39: string when() {
40:    timeval now;
41:    gettimeofday (&now, NULL);
42:    double start = start_time.tv_sec + start_time.tv_usec / 1.0e6;
43:    double dnow = now.tv_sec + now.tv_usec / 1.0e6;
44:    double delta = dnow - start;
45:    stringstream diff;
46:    diff << fixed << setw(8) << setprecision(5) << delta << " ";
47:    return diff.str();;
48: }
49:
```

```
50:
51: template <typename Type>
52: Type from_string (const string &str) {
53:    stringstream stream;
54:    stream << str;
55:    Type result;
56:    if (not (stream >> result and stream.eof())) {
57:       throw domain_error (string (typeid (Type).name())
58:             + " from_string (" + str + ")");
59:    }
60:    return result;
61: }
62:
63: void usage() {
64:    cerr << "Usage: " << progname << " children pdelay loops cdelays"
65:         << endl;
66:    exit (EXIT_FAILURE);
67: }
68:
69: void error (const string& obj) {
70:    errlog << obj << ": " << string (strerror (errno)) << endl;
71:    exit (EXIT_FAILURE);
72: }
73:
74: struct options {
75:    size_t children {};
76:    size_t pdelay {};
77:    size_t loops {};
78:    size_t cdelays {};
79:    options (const vector<string> opts) {
80:       if (opts.size() != 4) usage();
81:       try {
82:          children = from_string<size_t> (opts[0]);
83:          pdelay = from_string<size_t> (opts[1]);
84:          loops = from_string<size_t> (opts[2]);
85:          cdelays = from_string<size_t> (opts[3]);
86:       }catch (domain_error& error) {
87:          cerr << progname << ": " << error.what() << endl;
88:          usage();
89:       }
90:    }
91: };
92:
```

```
 93:
 94: pid_t wait_children() {
 95:    for (;;) {
 96:       int status;
 97:       pid_t child_pid = waitpid (-1, &status, WNOHANG);
 98:       if (child_pid <= 0) return child_pid;
 99:       outlog << when() << "Child pid " << child_pid
100:              << " status " << (status >> 8)
101:              << " signal " << (status & 0x7F)
102:              << " core " << (status >> 7 & 1) << endl;
103:    }
104: }
105:
106: int run_child (size_t id, size_t loops, size_t cdelays) {
107:    outlog << when() << "Child " << id << " starting." << endl;
108:    for (size_t loop = 0; loop < loops; ++loop) {
109:       sleep (cdelays);
110:       outlog << when() << "Child " << id << " message " << loop
111:              << "." << endl;
112:    }
113:    outlog << when() << "Child " << id << " finished." << endl;
114:    exit (id);
115:    // Must not return to main function.
116: }
117:
118: int main (int argc, char** argv) {
119:    progname = basename (argv[0]);
120:    outlog.set_execname (progname);
121:    errlog.set_execname (progname);
122:    vector<string> args (&argv[1], &argv[argc]);
123:    gettimeofday (&start_time, NULL);
124:    options opts (args);
125:    outlog << when() << "Main starting." << endl;
126:    for (size_t child = 0; child < opts.children; ++child) {
127:       pid_t pid = fork();
128:       if (pid < 0) error ("fork");
129:       if (pid == 0) run_child (child, opts.loops, opts.cdelays);
130:       outlog << when() << "Child " << pid << " created." << endl;
131:       wait_children();
132:       sleep (opts.pdelay);
133:    }
134:    for (;;) {
135:       outlog << when() << "Waiting for Godot." << endl;
136:       pid_t child_pid = wait_children();
137:       if (child_pid < 0) break;
138:       sleep (opts.pdelay);
139:    }
140:    outlog << when() << "Main finished." << endl;
141: }
142:
143: //TEST//
144:
145: //TEST// ./childwait 5 5 5 2 >childwait.out 2>&1
146: //TEST// mkpspdf childwait.ps childwait.cpp* childwait.out*
```

```
    1: @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ mkc: starting childwait.cpp
    2: childwait.cpp:
    3:       $Id: childwait.cpp,v 1.2 2014-05-23 12:26:39-07 - - $
    4: g++ -g -O0 -Wall -Wextra -std=gnu++11 childwait.cpp -o childwait -lglut
-lGLU -lGL -lX11 -lm -lrt
    5: rm -f childwait.o
    6: @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ mkc: finished childwait.cpp
```

```
 1: childwait(12644):  0.00902 Main starting.
 2: childwait(12644):  0.00932 Child 12645 created.
 3: childwait(12645):  0.00936 Child 0 starting.
 4: childwait(12645):  2.00971 Child 0 message 0.
 5: childwait(12645):  4.00983 Child 0 message 1.
 6: childwait(12644):  5.00966 Child 12646 created.
 7: childwait(12646):  5.00970 Child 1 starting.
 8: childwait(12645):  6.00994 Child 0 message 2.
 9: childwait(12646):  7.00995 Child 1 message 0.
10: childwait(12645):  8.01004 Child 0 message 3.
11: childwait(12646):  9.01006 Child 1 message 1.
12: childwait(12645): 10.01018 Child 0 message 4.
13: childwait(12645): 10.01033 Child 0 finished.
14: childwait(12674): 10.01020 Child 2 starting.
15: childwait(12646): 11.01022 Child 1 message 2.
16: childwait(12674): 12.01050 Child 2 message 0.
17: childwait(12646): 13.01034 Child 1 message 3.
18: childwait(12674): 14.01063 Child 2 message 1.
19: childwait(12646): 15.01054 Child 1 message 4.
20: childwait(12646): 15.01065 Child 1 finished.
21: childwait(12644): 15.01060 Child 12715 created.
22: childwait(12644): 15.01074 Child pid 12645 status 0 signal 0 core 0
23: childwait(12715): 15.01065 Child 3 starting.
24: childwait(12674): 16.01080 Child 2 message 2.
25: childwait(12715): 17.01101 Child 3 message 0.
26: childwait(12674): 18.01093 Child 2 message 3.
27: childwait(12715): 19.01113 Child 3 message 1.
28: childwait(12674): 20.01115 Child 2 message 4.
29: childwait(12674): 20.01121 Child 2 finished.
30: childwait(12644): 20.01129 Child pid 12646 status 1 signal 0 core 0
31: childwait(12716): 20.01121 Child 4 starting.
32: childwait(12715): 21.01125 Child 3 message 2.
33: childwait(12716): 22.01151 Child 4 message 0.
34: childwait(12715): 23.01137 Child 3 message 3.
35: childwait(12716): 24.01170 Child 4 message 1.
36: childwait(12715): 25.01151 Child 3 message 4.
37: childwait(12715): 25.01155 Child 3 finished.
38: childwait(12716): 26.01183 Child 4 message 2.
39: childwait(12716): 28.01197 Child 4 message 3.
40: childwait(12716): 30.01216 Child 4 message 4.
41: childwait(12716): 30.01221 Child 4 finished.
42: childwait(12644): 30.04338 Waiting for Godot.
43: childwait(12644): 30.04348 Child pid 12715 status 3 signal 0 core 0
44: childwait(12644): 35.04358 Waiting for Godot.
45: childwait(12644): 35.04372 Child pid 12716 status 4 signal 0 core 0
46: childwait(12644): 35.06292 Main finished.
```