

```
1
2  #include <stdlib.h>
3  #include <stdio.h>
4  #include <string.h>
5  #include <unistd.h>
6  #include <sys/wait.h>
7  #include <sys/time.h>
8
9  char ignore_errors = 0;
10 char verbose = 0;
11 char *argfile = NULL;
12
13 char error_seen = 0;
14
15 char **fa_argv;
16 int fa_argc;
17 void print_argv(char **args)
18 {
19     int i=0;
20     while(args[i]) printf("%s ",args[i++]);
21     printf("\n");
22 }
23
24 int set_opts(int argc, char** argv)
25 {
26     int i=1;
27     while(i<argc)
28     {
29         if(argv[i][0] == '-')
30         {
31             int pos = 1, len = strlen(argv[i]);
32             while(pos && pos < len)
33             {
34                 char o = argv[i][pos];
35                 //printf(" %c ",o);
36                 switch(o)
37                 {
38                     case 'i':
39                         ignore_errors = 1;
40                         pos++;
41                         break;
42                     case 'v':
43                         verbose = 1;
44                         pos++;
45                         break;
46                     case 'a':
47                         argfile = argv[++i];
```

```

48         pos=0;
49         break;
50     default:
51         printf("Unknown option %c\n",o);
52         exit(1);
53         break;
54     }
55 }
56 }
57 else return i;
58 i++;
59 }
60 return 0;
61 }
62
63 void read_file(char *argfile)
64 {
65     FILE *af = fopen(argfile,"r");
66     char **args = (char **)malloc(sizeof(char**)*64);
67     int argc = 0;
68     char buf[128];
69     while((fscanf(af,"%s",buf))!=EOF)
70     {
71         unsigned int size = strlen(buf);
72         char *arg = (char *)malloc(size);
73         strncpy(arg,buf,size);
74         args[argc++] = arg;
75     }
76     args[argc] = NULL;
77     fa_argv = args;
78     fa_argc = argc;
79 }
80
81
82
83 char **append_args(int argc, char** argv)
84 {
85     char **new_args = (char **)malloc(sizeof(char **) * (argc + fa_argc + 1));
86     //printf("%d %d %d\n",argc,fa_argc,argc+fa_argc);
87     memcpy(new_args, argv, argc*sizeof(char **));
88     memcpy(new_args + argc, fa_argv,fa_argc*sizeof(char**));
89     new_args[argc+fa_argc] = NULL;
90     return new_args;
91 }
92
93 int child_pid;
94 void timeout_handler(int a)
95 {
96     kill(child_pid,9);

```

```

97     printf("hihihi\n");
98 }
99
100 void milli_alarm(long int ms)
101 {
102     struct itimerval old, new;
103     new.it_interval.tv_usec = 0;
104     new.it_interval.tv_sec = 0;
105     new.it_value.tv_sec = 0;
106     new.it_value.tv_usec = 1000*ms;
107     setitimer(ITIMER_VIRTUAL, &new, NULL);
108 }
109
110 FILE *f;
111 int main(int argc, char** argv, char *envp[])
112 {
113     int a;
114     f = stdin;
115     if((a = set_opts(argc, argv)) f = fopen(argv[a], "r");
116
117     if(argfile != NULL) read_file(argfile);
118
119     //printf("%d %d %s\n", ignore_errors, verbose, argfile);
120     char input[128];
121     int stat = 0;
122     while(fgets(input, sizeof(input), f) != NULL)
123     {
124         unsigned int j = 0;
125         input[strlen(input)-1] = '\0';
126         if(input[0] != '#')
127         {
128             char* toks[64];
129             char* token;
130             token = strtok(input, " ");
131             while(token != NULL)
132             {
133                 toks[j++] = token;
134                 token = strtok(NULL, " ");
135             }
136             toks[j] = NULL;
137
138             char **args;
139             if(argfile != NULL) args = append_args(j, toks);
140             else args = toks;
141
142             if(verbose) print_argv(args);
143
144             switch(child_pid = fork())
145             {

```

```
146     case -1:
147         exit(1);
148     case 0:
149         stat = execvp(args[0],args);
150         printf("hello\n");
151         exit(stat);
152     default:
153         signal(SIGALRM,timeout_handler);
154         milli_alarm(1);
155         waitpid(child_pid,&stat,0);
156         printf("done\n");
157         if(argfile != NULL) free(args);
158     }
159
160     int exitstat = WEXITSTATUS(stat);
161     if (exitstat)
162     {
163         if(ignore_errors) error_seen = 1;
164         else
165         {
166             if(exitstat == 255) exit(1);
167             else exit(1);
168         }
169     }
170 }
171 }
172 return 0;
173 }
```