```c

#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <unistd.h>
#include <sys/wait.h>
#include <sys/time.h>

char   ignore_errors   = 0;
char   verbose        = 0;
char   *argfile      = NULL;

char   error_seen      = 0;

char   **fa_argv;
int      fa_argc;
void print_argv(char **args)
{
   int i=0;
   while(args[i]) printf("%s ",args[i++]);
   printf("\n");
}

int set_opts(int argc, char** argv)
{
   int i=1;
   while(i<argc)
   {
      if(argv[i][0] == '-')
      {
         int pos = 1, len = strlen(argv[i]);
         while(pos && pos < len)
         {
            char o = argv[i][pos];
            //printf(" %c ",o);
            switch(o)
            {
               case 'i':
                  ignore_errors = 1;
                  pos++;
                  break;
               case 'v':
                  verbose = 1;
                  pos++;
                  break;
               case 'a':
                  argfile = argv[++i];
                  pos=0;
                  break;
               default:
                  printf(”Unknown option %c\n”,o);
                  exit(1);
                  break;
            }
         }
      }
      else return i;
      i++;
   }
   return 0;
}

void read_file(char *argfile)
{
   FILE *af = fopen(argfile,”r”);
   char **args = (char **)malloc(sizeof(char**)*64);
   int argc = 0;
   char buf[128];
   while((fscanf(af,”%s”,buf))!=EOF)
   {
      unsigned int size = strlen(buf);
      char *arg = (char *)malloc(size);
      strncpy(arg,buf,size);
      args[argc++] = arg;
   }
   args[argc] = NULL;
   fa_argv = args;
   fa_argc = argc;
}



char **append_args(int argc, char** argv)
{
   char **new_args = (char **)malloc(sizeof(char **) * (argc + fa_argc + 1));
   //printf(”%d %d %d\n”,argc,fa_argc,argc+fa_argc);
   memcpy(new_args,      argv,   argc*sizeof(char **));
   memcpy(new_args + argc,   fa_argv,fa_argc*sizeof(char**));
   new_args[argc+fa_argc] = NULL;
   return new_args;
}

int child_pid;
void timeout_handler(int a)
{
   kill(child_pid,9);
```

```c
   printf("hihihi\n");
}

void milli_alarm(long int ms)
{
   struct itimerval old, new;
   new.it_interval.tv_usec = 0;
   new.it_interval.tv_sec = 0;
   new.it_value.tv_sec = 0;
   new.it_value.tv_usec = 1000*ms;
   setitimer(ITIMER_VIRTUAL, &new,NULL);
}

FILE *f;
int main(int argc,char** argv,char *envp[])
{
   int a;
   f = stdin;
   if((a = set_opts(argc,argv))) f = fopen(argv[a],"r");

   if(argfile != NULL) read_file(argfile);

   //printf("%d %d %s\n",ignore_errors,verbose,argfile);
   char input[128];
   int stat = 0;
   while(fgets(input,sizeof(input),f)!=NULL)
   {
      unsigned int j = 0;
      input[strlen(input)-1] = '\0';
      if(input[0] !='#')
      {
         char* toks[64];
         char* token;
         token = strtok(input," ");
         while(token !=NULL)
         {
            toks[j++] = token;
            token = strtok(NULL," ");
         }
         toks[j] = NULL;

         char **args;
         if(argfile != NULL) args = append_args(j,toks);
         else args = toks;

         if(verbose) print_argv(args);

         switch(child_pid = fork())
         {
            case -1:
               exit(1);
            case 0:
               stat = execvp(args[0],args);
               printf(”hello\n”);
               exit(stat);
            default:
               signal(SIGALRM,timeout_handler);
               milli_alarm(1);
               waitpid(child_pid,&stat,0);
               printf(”done\n”);
               if(argfile != NULL) free(args);
         }

         int exitstat = WEXITSTATUS(stat);
         if (exitstat)
         {
            if(ignore_errors) error_seen = 1;
            else
            {
               if(exitstat == 255) exit(1);
               else exit(1);
            }
         }
      }
   }
   return 0;
}
```