C-net

Presented by

Bruk Zewdie, Kidus Mulu, Kingsley Neequaye, Rediet Bekele,
William Oseghare

HELLO!

The C-net team

Bruk Zewdie
—System Architect

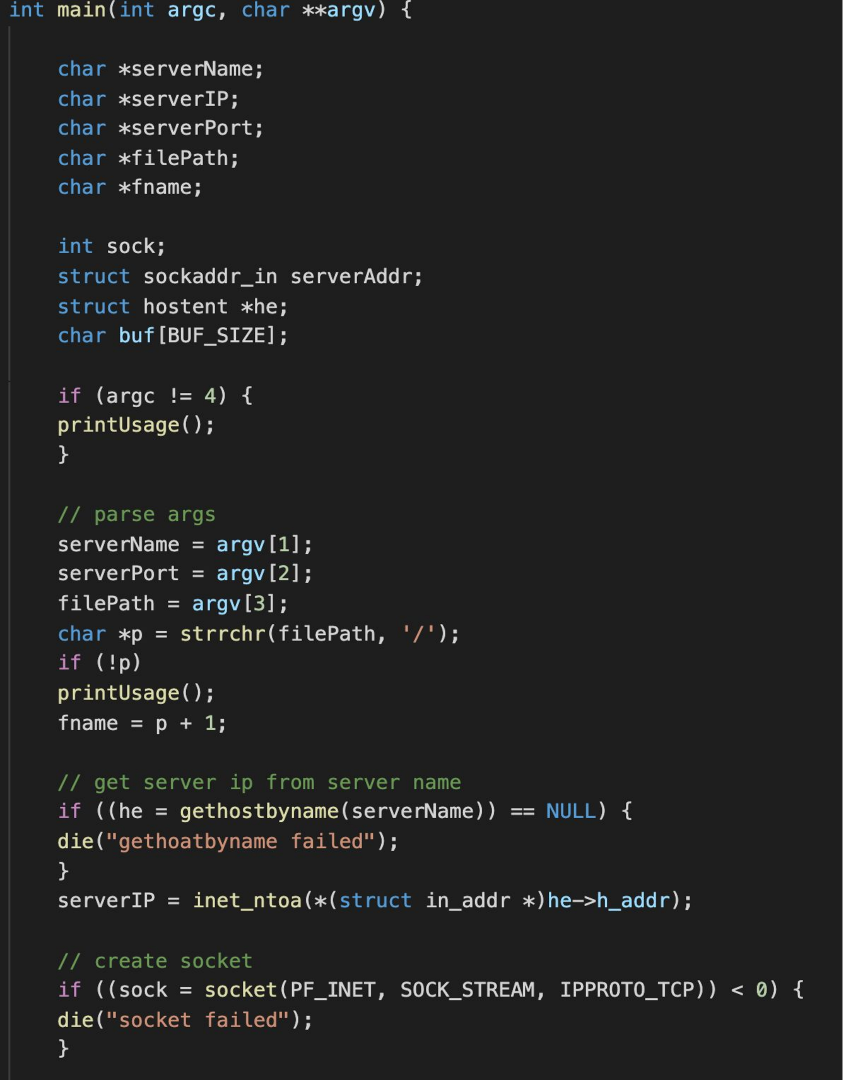
Kidus Mulu
—Language Guru

Kingsley Neequaye
—Tester

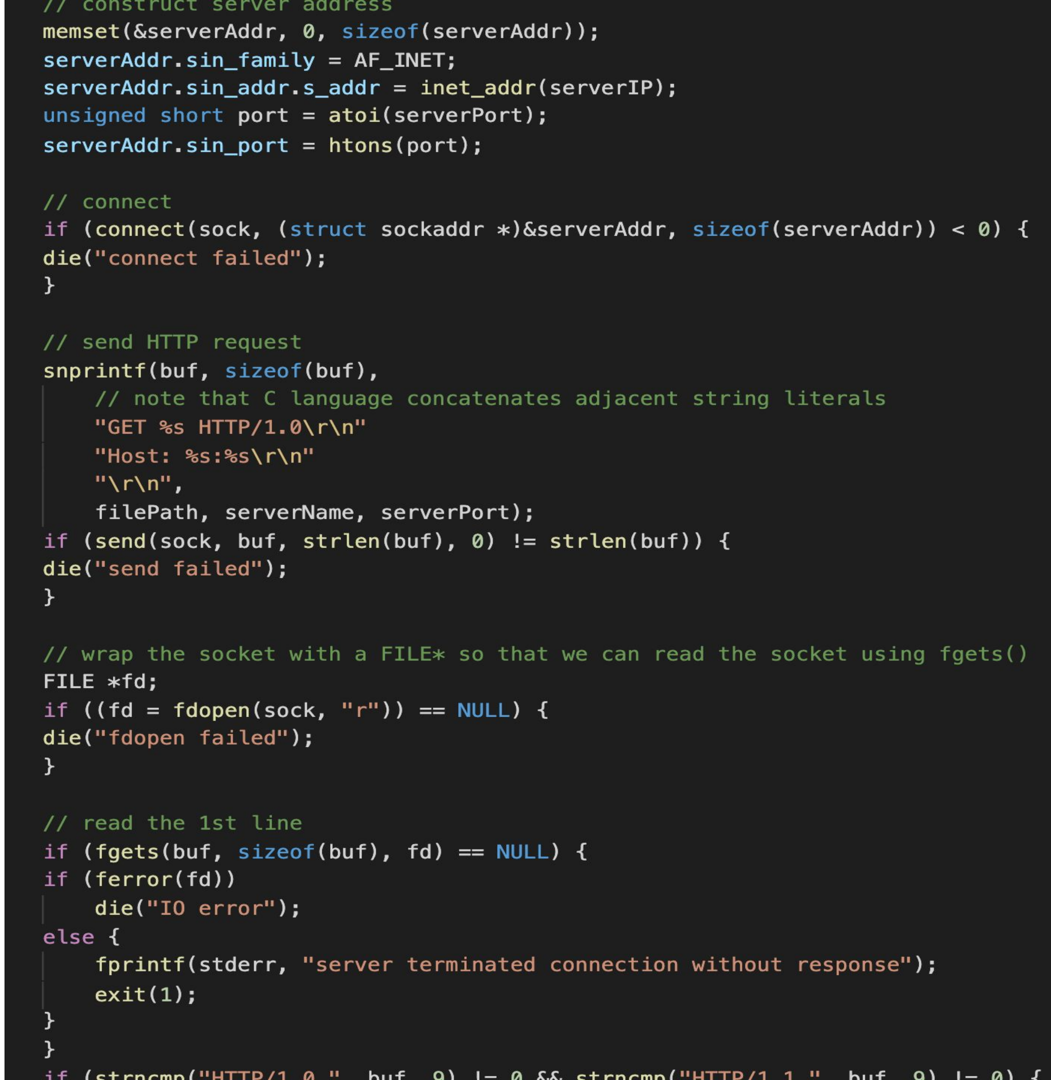
Rediet Bekele
—Manager

William Oseghare
—System Architect

WHAT IS C-NET?



```c
int main(int argc, char **argv) {

    char *serverName;
    char *serverIP;
    char *serverPort;
    char *filePath;
    char *fname;

    int sock;
    struct sockaddr_in serverAddr;
    struct hostent *he;
    char buf[BUF_SIZE];

    if (argc != 4) {
    printUsage();
    }

    // parse args
    serverName = argv[1];
    serverPort = argv[2];
    filePath = argv[3];
    char *p = strrchr(filePath, '/');
    if (!p)
    printUsage();
    fname = p + 1;

    // get server ip from server name
    if ((he = gethostbyname(serverName)) == NULL) {
    die("gethoatbyname failed");
    }
    serverIP = inet_ntoa(*(struct in_addr *)he->h_addr);

    // create socket
    if ((sock = socket(PF_INET, SOCK_STREAM, IPPROTO_TCP)) < 0) {
    die("socket failed");
    }
```



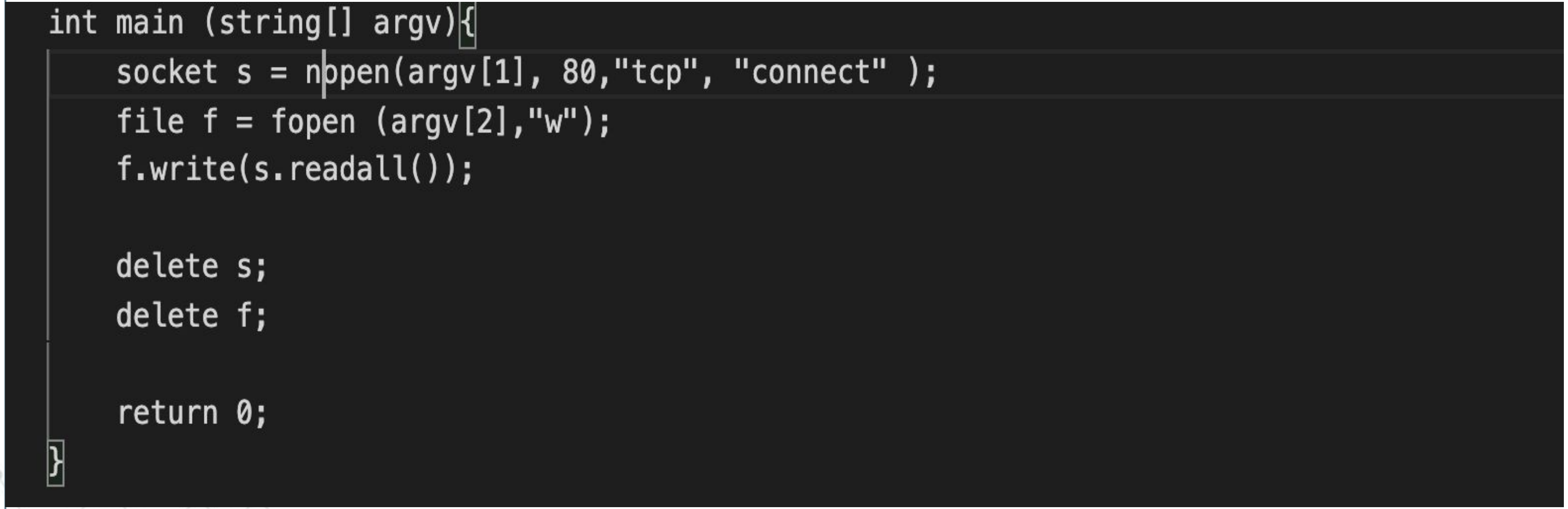
```c
// construct server address
memset(&serverAddr, 0, sizeof(serverAddr));
serverAddr.sin_family = AF_INET;
serverAddr.sin_addr.s_addr = inet_addr(serverIP);
unsigned short port = atoi(serverPort);
serverAddr.sin_port = htons(port);

// connect
if (connect(sock, (struct sockaddr *)&serverAddr, sizeof(serverAddr)) < 0) {
die("connect failed");
}

// send HTTP request
snprintf(buf, sizeof(buf),
    // note that C language concatenates adjacent string literals
    "GET %s HTTP/1.0\r\n"
    "Host: %s:%s\r\n"
    "\r\n",
    filePath, serverName, serverPort);
if (send(sock, buf, strlen(buf), 0) != strlen(buf)) {
die("send failed");
}

// wrap the socket with a FILE* so that we can read the socket using fgets()
FILE *fd;
if ((fd = fdopen(sock, "r")) == NULL) {
die("fdopen failed");
}

// read the 1st line
if (fgets(buf, sizeof(buf), fd) == NULL) {
if (ferror(fd))
    die("IO error");
else {
    fprintf(stderr, "server terminated connection without response");
    exit(1);
}
}
if (strncmp("HTTP/1.0 ", buf, 9) != 0 && strncmp("HTTP/1.1 ", buf, 9) != 0) {
```

WHAT IS C-NET?

A language for network programming based on C that **provides a simple way for programmers to code network/file programs** through succinct code and simple manipulation of files and sockets

WHAT IS C-NET?

Motivation



```
int main (string[] argv){
    socket s = nopen(argv[1], 80,"tcp", "connect" );
    file f = fopen (argv[2],"w");
    f.write(s.readall());

    delete s;
    delete f;

    return 0;
}
```

Motivation and Core Features

Motivation

- Simplify I/O management for socket programming
- Provide intuitive interface for using strings
- Streamline implementation of network/file programs
- Discard complex dynamic memory management interface of C

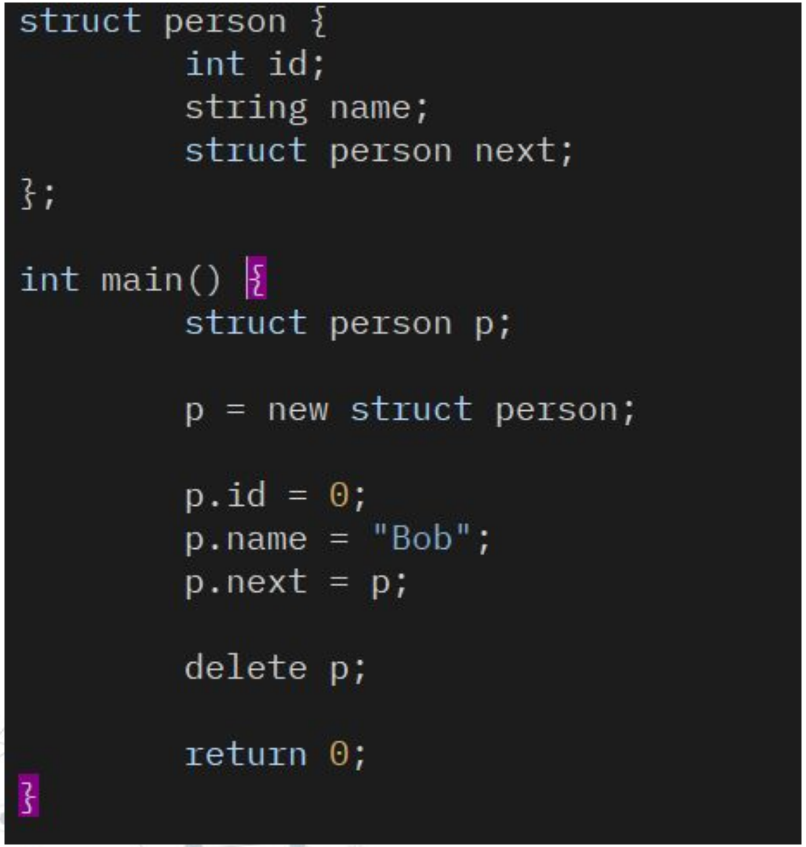
C-net core features:

- IO implementation
- String implementation
- C-net standard library
- Streamlined memory management

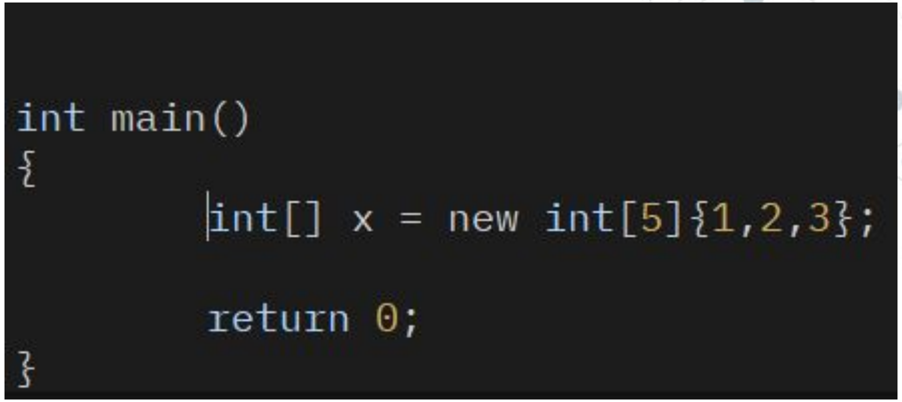
WHAT ARE THE FEATURES?

Array and Structs

- Arrays can be declared and initialized using Java like syntax
- Arrays can hold any type of data, even structs and other arrays
- Struct members themselves can be structs
- Struct declaration and use is based on C like syntax, but abstracts away manual memory allocation
- All access is treated as pointer access, so there is only the . operator



```
struct person {
        int id;
        string name;
        struct person next;
};

int main() {
        struct person p;

        p = new struct person;

        p.id = 0;
        p.name = "Bob";
        p.next = p;

        delete p;

        return 0;
}
```



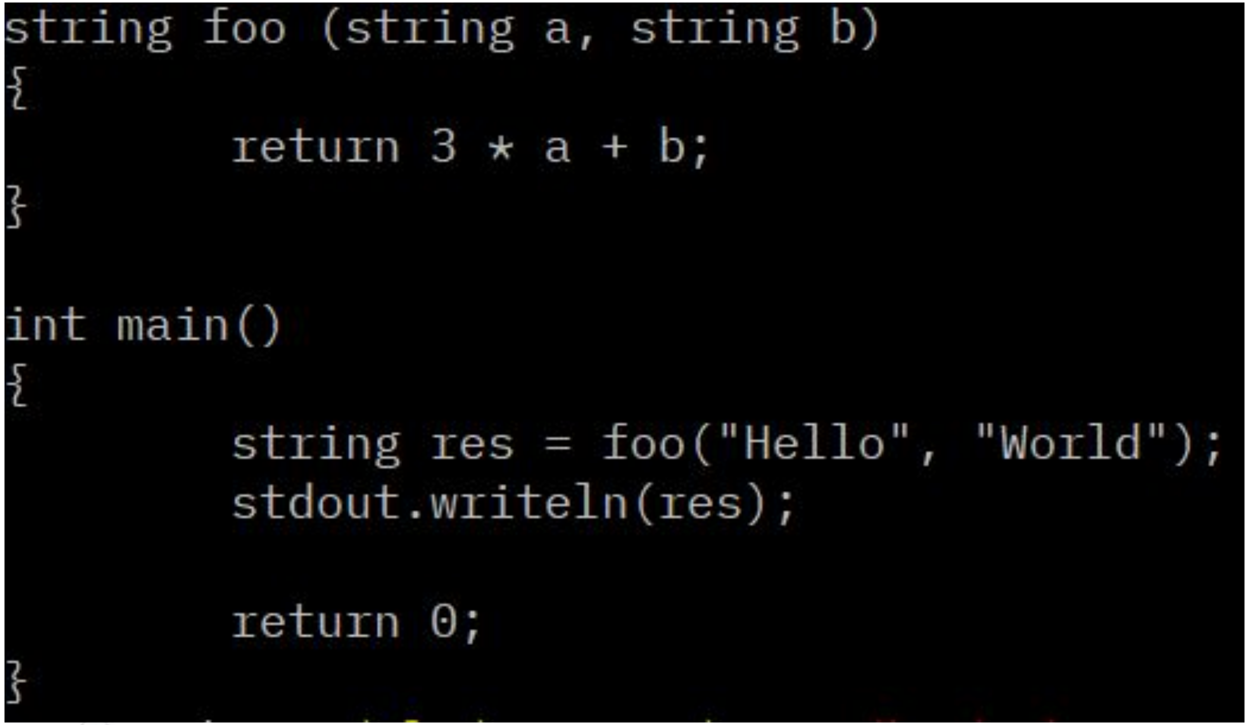
```
int main()
{
        int[] x = new int[5]{1,2,3};

        return 0;
}
```

String Implementation

- Heap allocated, immutable strings
    - As a result, a string is only relevant in its declared block, with few exceptions
- Everything is automated from creation to deletion, including managing temporary strings for string operations
- Used by all read/write I/O operations throughout Cnet

Examples

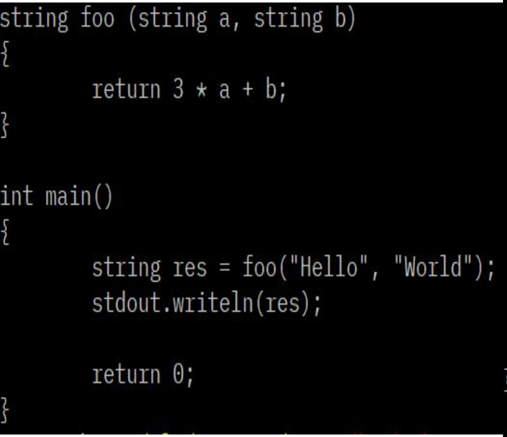
User program



```
string foo (string a, string b)
{
        return 3 * a + b;
}

int main()
{
        string res = foo("Hello", "World");
        stdout.writeln(res);

        return 0;
}
```

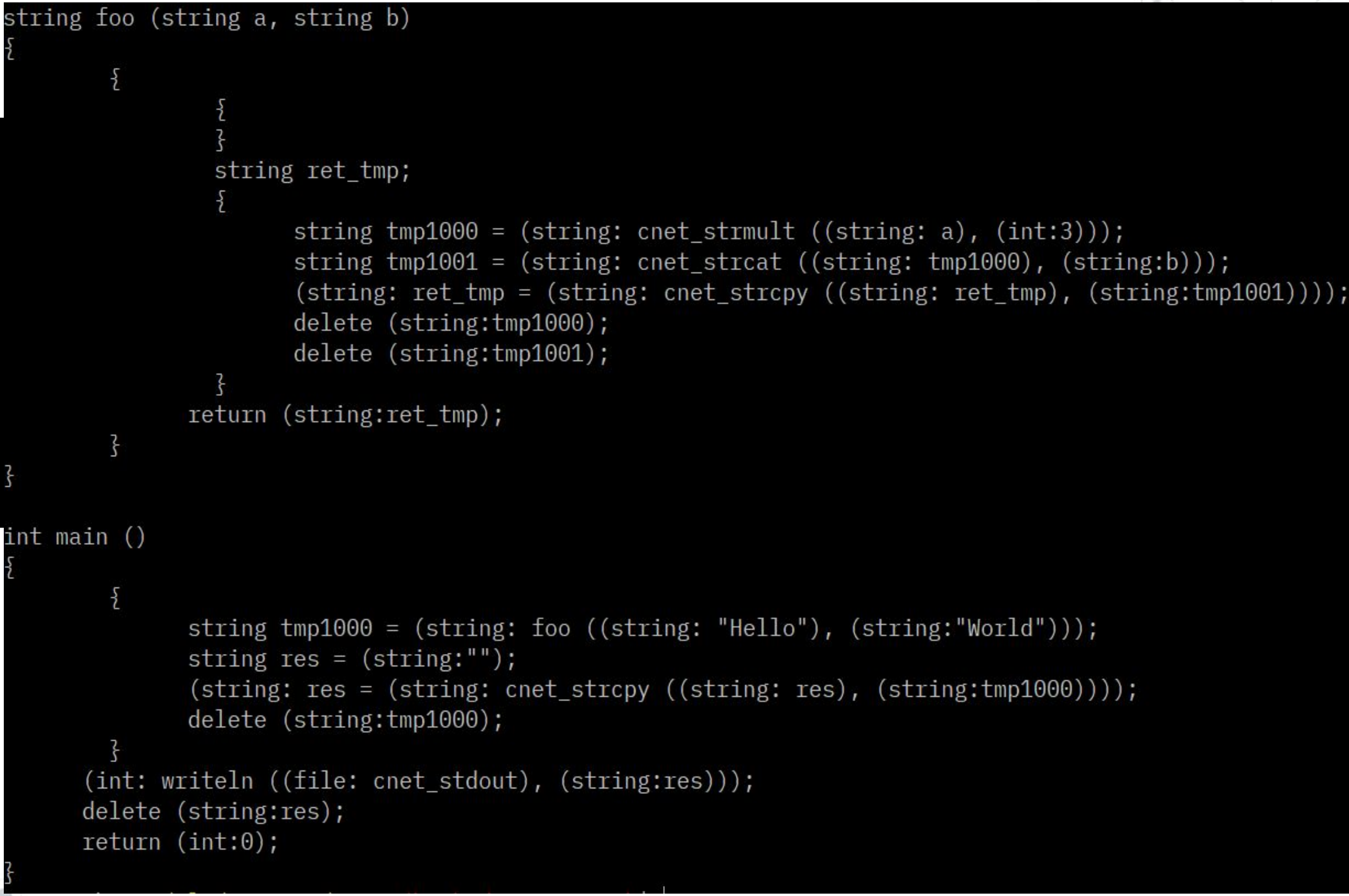
Examples

Source Code



```
string foo (string a, string b)
{
        return 3 * a + b;
}

int main()
{
        string res = foo("Hello", "World");
        stdout.writeln(res);

        return 0;
}
```

Codegen code

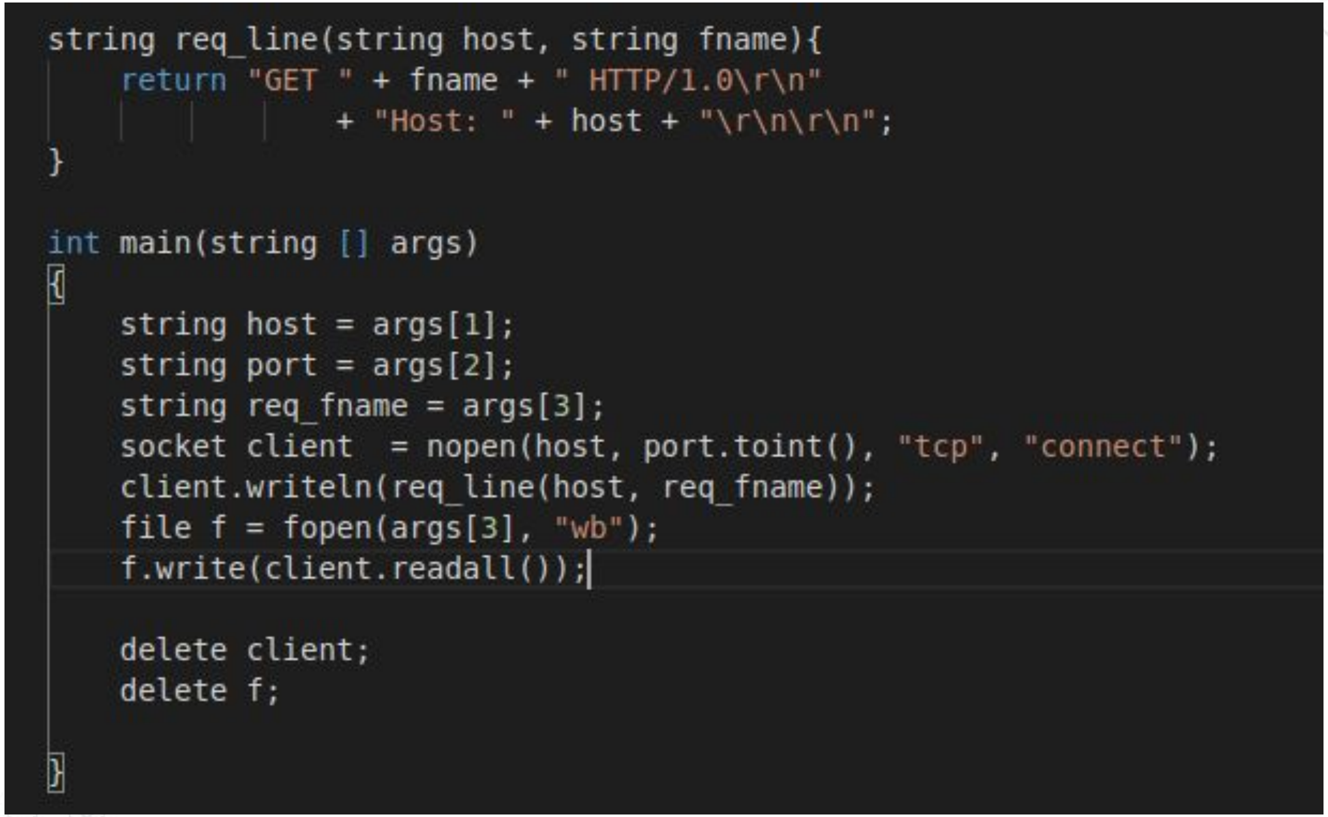


```
string foo (string a, string b)
{
        {
                {
                }
                string ret_tmp;
                {
                        string tmp1000 = (string: cnet_strmult ((string: a), (int:3)));
                        string tmp1001 = (string: cnet_strcat ((string: tmp1000), (string:b)));
                        (string: ret_tmp = (string: cnet_strcpy ((string: ret_tmp), (string:tmp1001))));
                        delete (string:tmp1000);
                        delete (string:tmp1001);
                }
              return (string:ret_tmp);
        }
}

int main ()
{
        {
              string tmp1000 = (string: foo ((string: "Hello"), (string:"World")));
              string res = (string:"");
              (string: res = (string: cnet_strcpy ((string: res), (string:tmp1000))));
              delete (string:tmp1000);
        }
      (int: writeln ((file: cnet_stdout), (string:res)));
      delete (string:res);
      return (int:0);
}
```

IO Implementation

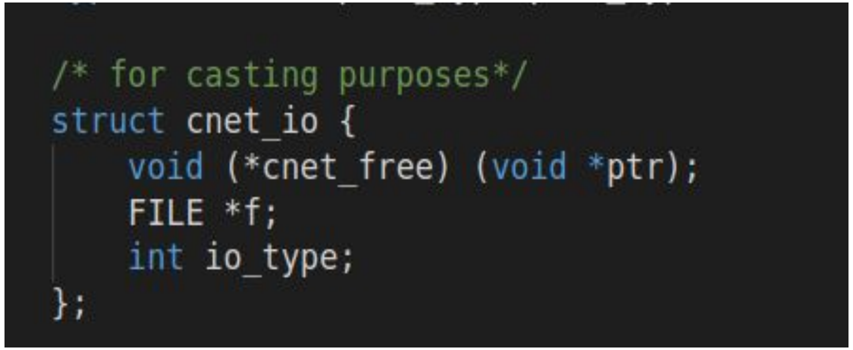
- I/O Interface in C-net standard library provides streamlined file/network programming implementation.
- Sockets and files are implemented polymorphically so that user can read/write to files and socket in the same manner.
- Error handling and diagnostic messages for file access.

Sample code: Client

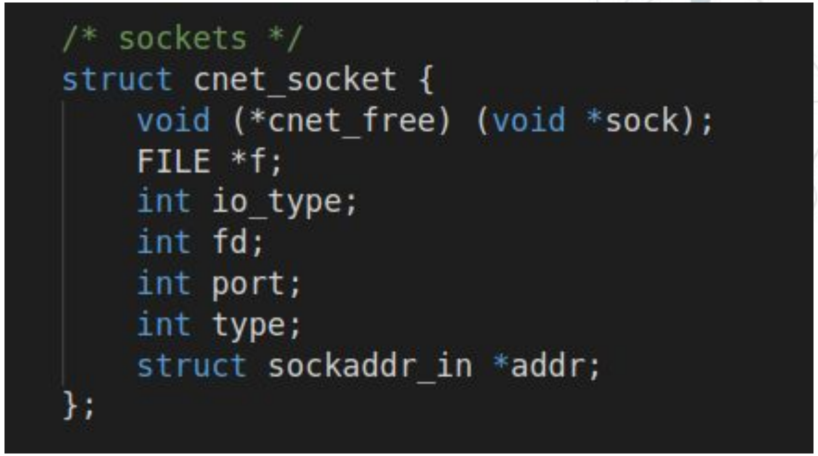


```
string req_line(string host, string fname){
    return "GET " + fname + " HTTP/1.0\r\n"
                + "Host: " + host + "\r\n\r\n";
}

int main(string [] args)
{
    string host = args[1];
    string port = args[2];
    string req_fname = args[3];
    socket client  = nopen(host, port.toint(), "tcp", "connect");
    client.writeln(req_line(host, req_fname));
    file f = fopen(args[3], "wb");
    f.write(client.readall());

    delete client;
    delete f;

}
```

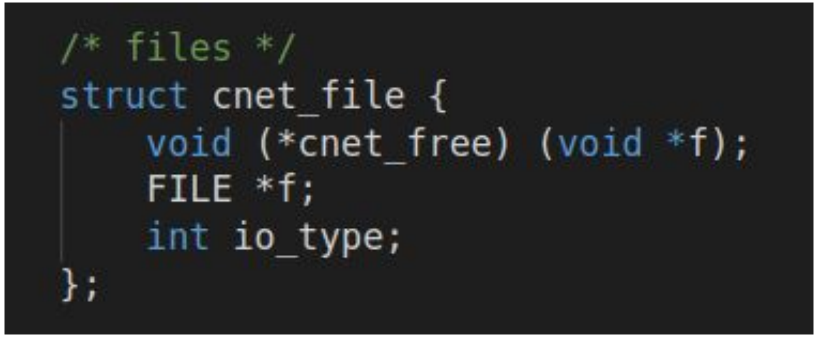
IO representation



```
/* for casting purposes*/
struct cnet_io {
    void (*cnet_free) (void *ptr);
    FILE *f;
    int io_type;
};
```

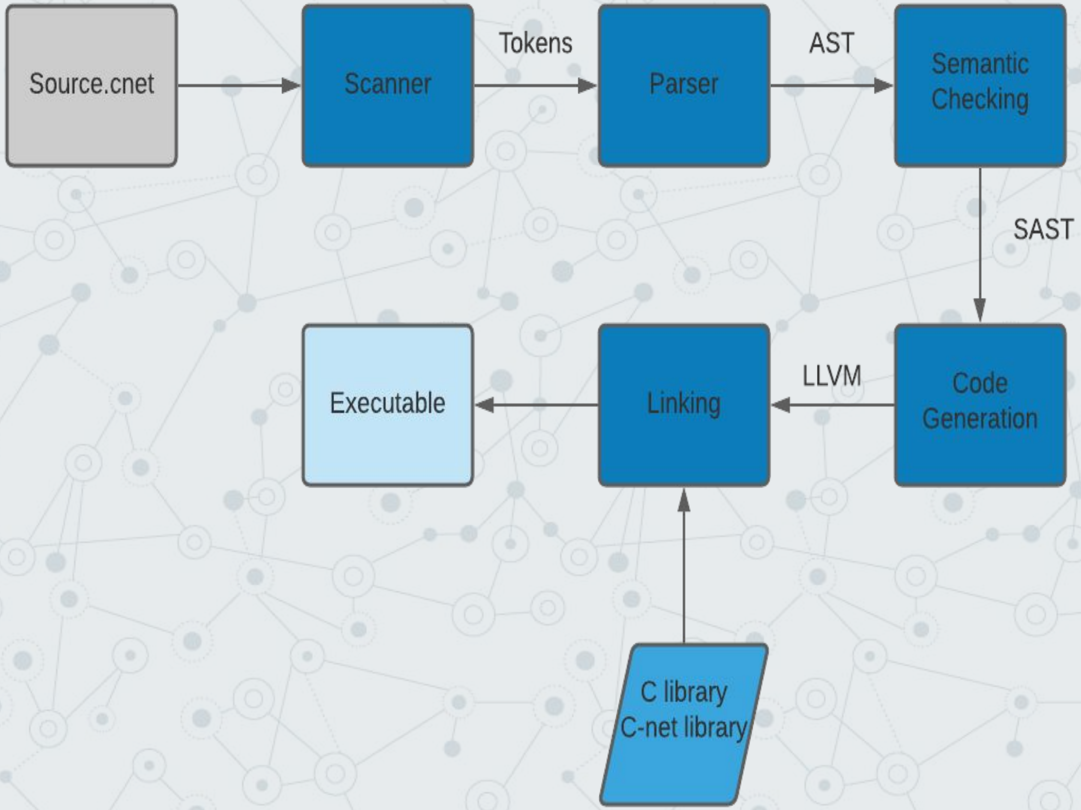


```
/* sockets */
struct cnet_socket {
    void (*cnet_free) (void *sock);
    FILE *f;
    int io_type;
    int fd;
    int port;
    int type;
    struct sockaddr_in *addr;
};
```



```
/* files */
struct cnet_file {
    void (*cnet_free) (void *f);
    FILE *f;
    int io_type;
};
```

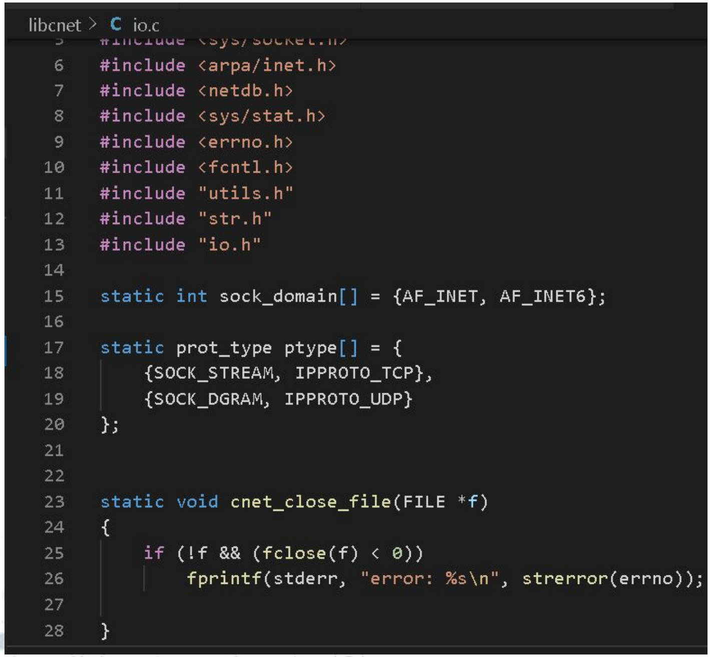
COMPILER ARCHITECTURE



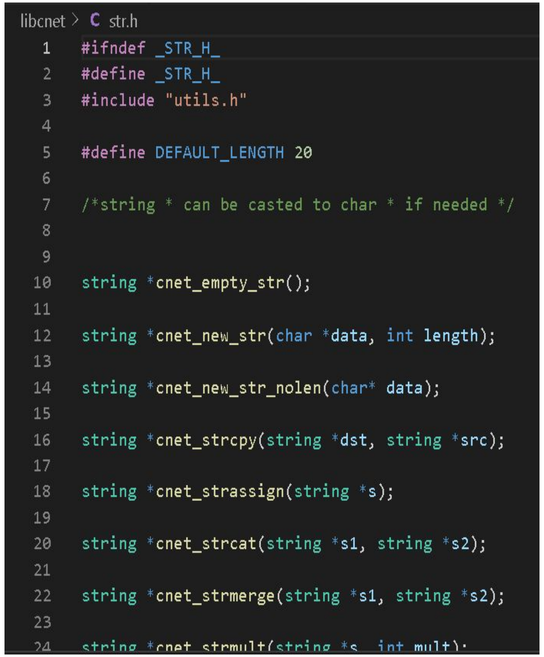
C-net source programming file (<filename>.cnet) completes is **compiled down into LLVM IR and linked in with the C-library and C-net library to produce a target executable**

STANDARD LIBRARY

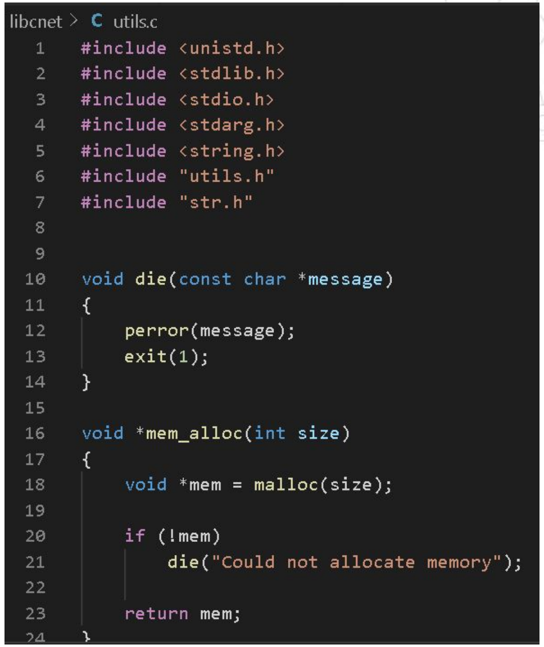
C-libraries in libcnet/ for implementing socket/file, string, and utility functions



```c
#include <sys/socket.h>
#include <arpa/inet.h>
#include <netdb.h>
#include <sys/stat.h>
#include <errno.h>
#include <fcntl.h>
#include "utils.h"
#include "str.h"
#include "io.h"

static int sock_domain[] = {AF_INET, AF_INET6};

static prot_type ptype[] = {
    {SOCK_STREAM, IPPROTO_TCP},
    {SOCK_DGRAM, IPPROTO_UDP}
};


static void cnet_close_file(FILE *f)
{
    if (!f && (fclose(f) < 0))
        fprintf(stderr, "error: %s\n", strerror(errno));

}
```



```c
#ifndef _STR_H_
#define _STR_H_
#include "utils.h"

#define DEFAULT_LENGTH 20

/*string * can be casted to char * if needed */


string *cnet_empty_str();

string *cnet_new_str(char *data, int length);

string *cnet_new_str_nolen(char* data);

string *cnet_strcpy(string *dst, string *src);

string *cnet_strassign(string *s);

string *cnet_strcat(string *s1, string *s2);

string *cnet_strmerge(string *s1, string *s2);

string *cnet_strmult(string *s, int mult);
```



```c
#include <unistd.h>
#include <stdlib.h>
#include <stdio.h>
#include <stdarg.h>
#include <string.h>
#include "utils.h"
#include "str.h"


void die(const char *message)
{
    perror(message);
    exit(1);
}

void *mem_alloc(int size)
{
    void *mem = malloc(size);

    if (!mem)
        die("Could not allocate memory");

    return mem;
}
```

Testing and Automation

Test Suite

- **Tests broken down by topic into subdirectories:**
  - Scanner/
  - Parser/
  - Semant/
  - Integration/
  - Stdlib/
- **Checks *.out against for output for test-*.cnet files, which are expected to pass**
- **Checks *.err against output for fail-*.cnet files, which are expected to fail**

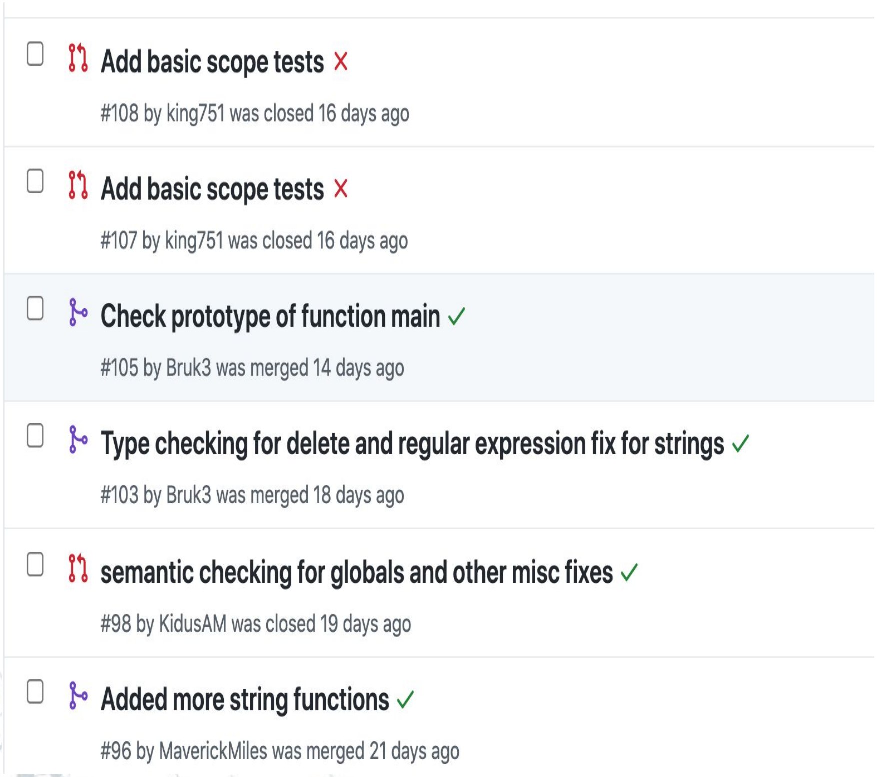
Tests Plan

- The modularization of tests has been extremely helpful in pinpointing exactly where a bug lies
- If given commit passes all of the tests for Scanner, Parser and Semant but fails on an integration test: Check Codegen or Stdlib!
- Integration tests reflect programs a C-net user may write

Testing and Automation

- The testing architecture is not limited to our local environment
- We used Github Actions to automate our development workflows
- On every PR, regression tests are run on a remote containerized environment that's hosted on Github Actions
- We get a status notification on our slack channel for the plt project

Testing and Automation



Add basic scope tests ✗
#108 by king751 was closed 16 days ago

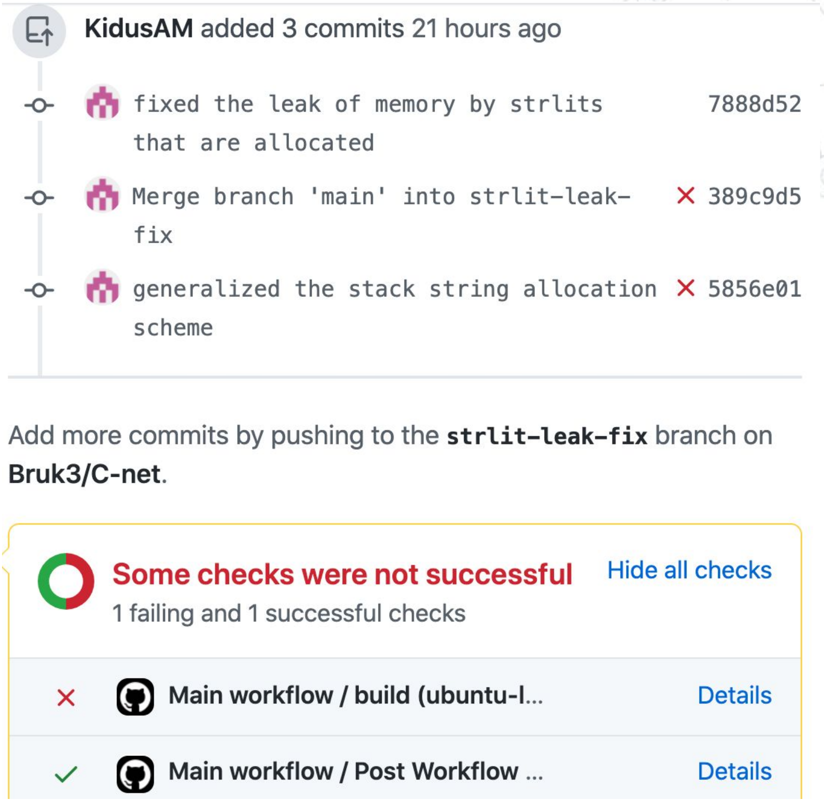
Add basic scope tests ✗
#107 by king751 was closed 16 days ago

Check prototype of function main ✓
#105 by Bruk3 was merged 14 days ago

Type checking for delete and regular expression fix for strings ✓
#103 by Bruk3 was merged 18 days ago

semantic checking for globals and other misc fixes ✓
#98 by KidusAM was closed 19 days ago

Added more string functions ✓
#96 by MaverickMiles was merged 21 days ago



KidusAM added 3 commits 21 hours ago

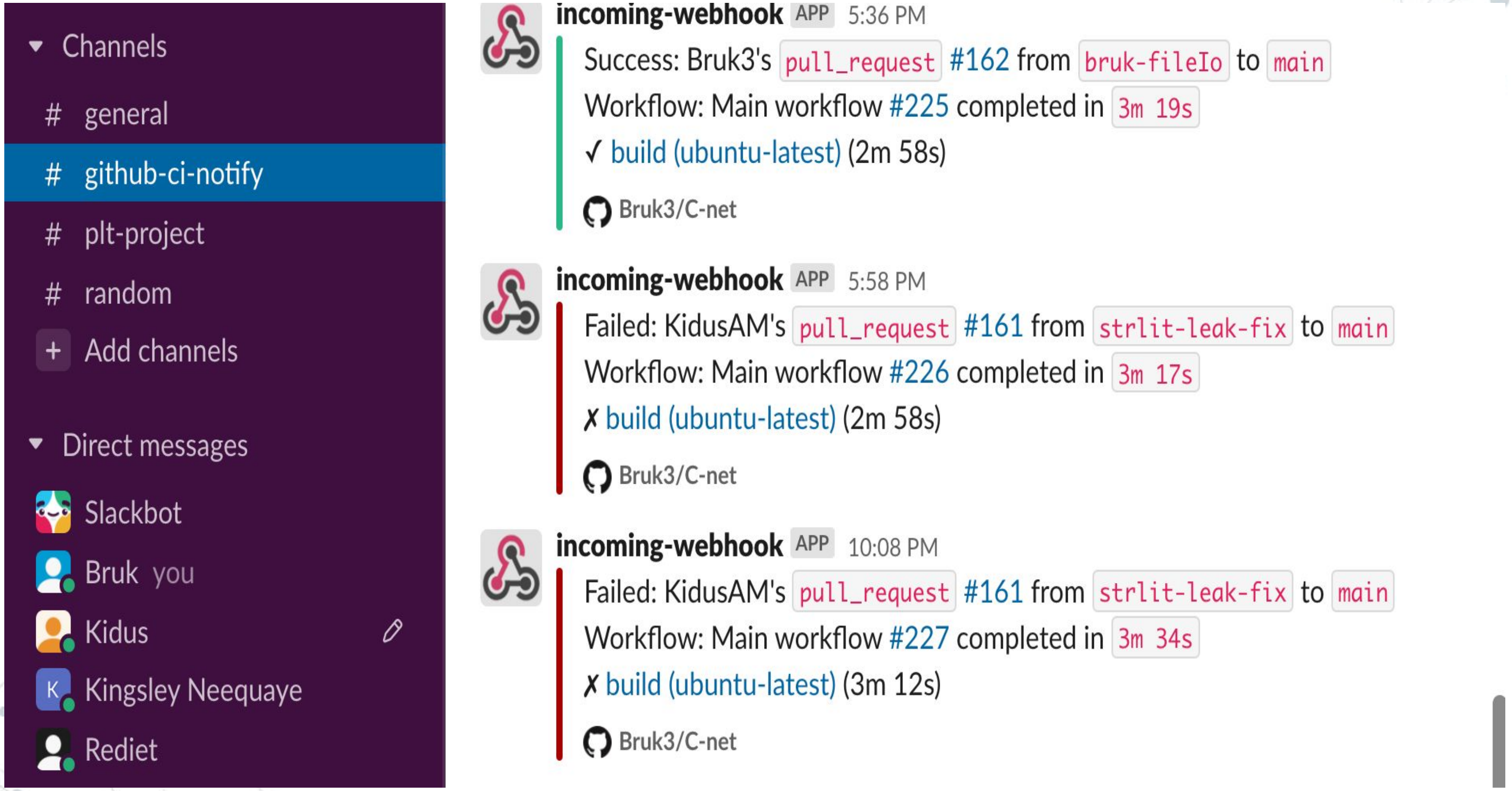
```
fixed the leak of memory by strlits          7888d52
that are allocated
Merge branch 'main' into strlit-leak-      ✗ 389c9d5
fix
generalized the stack string allocation    ✗ 5856e01
scheme
```

Add more commits by pushing to the **strlit-leak-fix** branch on **Bruk3/C-net**.

Some checks were not successful Hide all checks
1 failing and 1 successful checks

✗ Main workflow / build (ubuntu-l... Details
✓ Main workflow / Post Workflow ... Details

Testing and Automation



incoming-webhook APP 5:36 PM

Success: Bruk3's `pull_request` #162 from `bruk-fileIo` to `main`
Workflow: Main workflow #225 completed in `3m 19s`
✓ build (ubuntu-latest) (2m 58s)

Bruk3/C-net

incoming-webhook APP 5:58 PM

Failed: KidusAM's `pull_request` #161 from `strlit-leak-fix` to `main`
Workflow: Main workflow #226 completed in `3m 17s`
✗ build (ubuntu-latest) (2m 58s)

Bruk3/C-net

incoming-webhook APP 10:08 PM

Failed: KidusAM's `pull_request` #161 from `strlit-leak-fix` to `main`
Workflow: Main workflow #227 completed in `3m 34s`
✗ build (ubuntu-latest) (3m 12s)

Bruk3/C-net

DEMO!

2 Interesting C-net programs