# Distributed Systems
# Lecture 4

# Programming Interfaces for Internet Addresses

Joseph Phillips

Last modified 2019 October 25

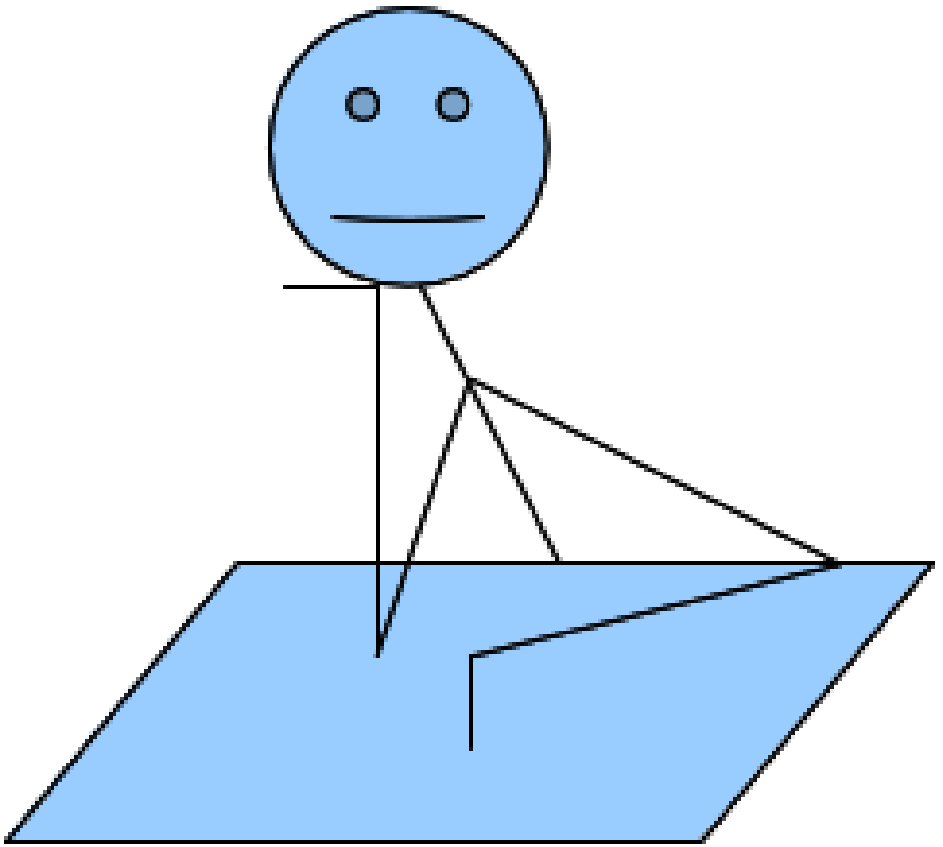
# Topics

- Call the DNS!
  - IP addresses/ hostnames
  - In Java
  - In C
- URIs and URLs
- Downloading webpages
  - In Java
  - In C with libcurl

# Motivation

"*Boy!  We sure learned some useful things about networking last lecture!*"

"*Now if only there was a way to apply that knowledge in a program . . .*"



# First some terminology . . .

- A ***node***
  - A device on a network
- A ***host***
  - Nodes that are computers
- An ***IP address***
  - is a number representing a host
  - IPv4 e.g.: 172.217.1.36
  - IPv6 e.g.: 2001:0db8:0000:0000:0000:8a2e:0370:7334
- A ***hostname***
  - DNS-assigned human-readable form of IP address
  - e.g. `www.depaul.edu`

# hostname => IP address (Java)

- import java.net.*;
- Class *`InetAddress`*
  - Represents both IPv4 and IPv6 IP addresses
- *`InetAddress.getByName()`* factory method:
  - InetAddress address = InetAddress.getByName("www.depaul.edu");
- *`InetAddress.getAllByName()`* factory method:

  ```
  InetAddress[ ] addresses =
  InetAddress.getAllByName("www.depaul.edu");

  for (InetAddress addr : addresses) { /* whatever processing */ }
  ```

- Both throw *`UnknownHostException`* if DNS cannot find name

# hostname => IP address (Java)

```
import  java.net.*;

public class DePaulByName
{
  public static void main       (String[]       args)
  {
    try
    {
      InetAddress       address = InetAddress.getByName("www.depaul.edu");
      System.out.println(address);
    }
    catch  (UnknownHostException ex)
    {
      System.out.println("Could not find www.depaul.edu");
    }
  }
}
```

# Your turn!

Write a Java program that prints the IP address of the hostname provided as `args[0]`.

# IP address => hostname (Java)

- *`InetAddress.getByName()`* does that too!

```
import  java.net.*;

public class DePaulByIpAddr
{
  public static void main       (String[]       args)
  {
    try
    {
      address           = InetAddress.getByName("216.220.178.116");
      System.out.println(address.getHostName());
    }
    catch  (UnknownHostException ex)
    {
      System.out.println("Could not find www.depaul.edu");
    }
  }
}
```

# Creating IP address from raw bytes

- *`InetAddress.getByAddress(byte[ ] byteArray)`*

  ```
  byte[ ] byteArray = {107, 23, (byte)216, (byte)196};
  InetAddress addr = InetAddress.getByAddress(byteArray);
  ```

  - Remember!  Java does not have an unsigned byte type
    - Must cast (byte)num if num in [128..255]
- Throws UnknownHostException only if byteArray has illegal size
  - neither 4 nor 16
  - Does not contact DNS

# “Which machine am *I* on?”

- *`InetAddress.getLocalHost()`*
  - Asks DNS for IP address
  - If unable to asks, then just returns “loopback” address:
    - localhost/127.0.0.1

# More methods on InetAddress

- *`public String getHostName()`*
  - Returns human-readable hostname
  - First relies on cached name, then DNS
- *`public String getCanonicalHostName()`*
  - First relies on DNS, then cached name
- *`public byte[] getAddress()`*
- *`public String getHostAddress()`*
  - Returns bytes of getAddress() in dotted notation

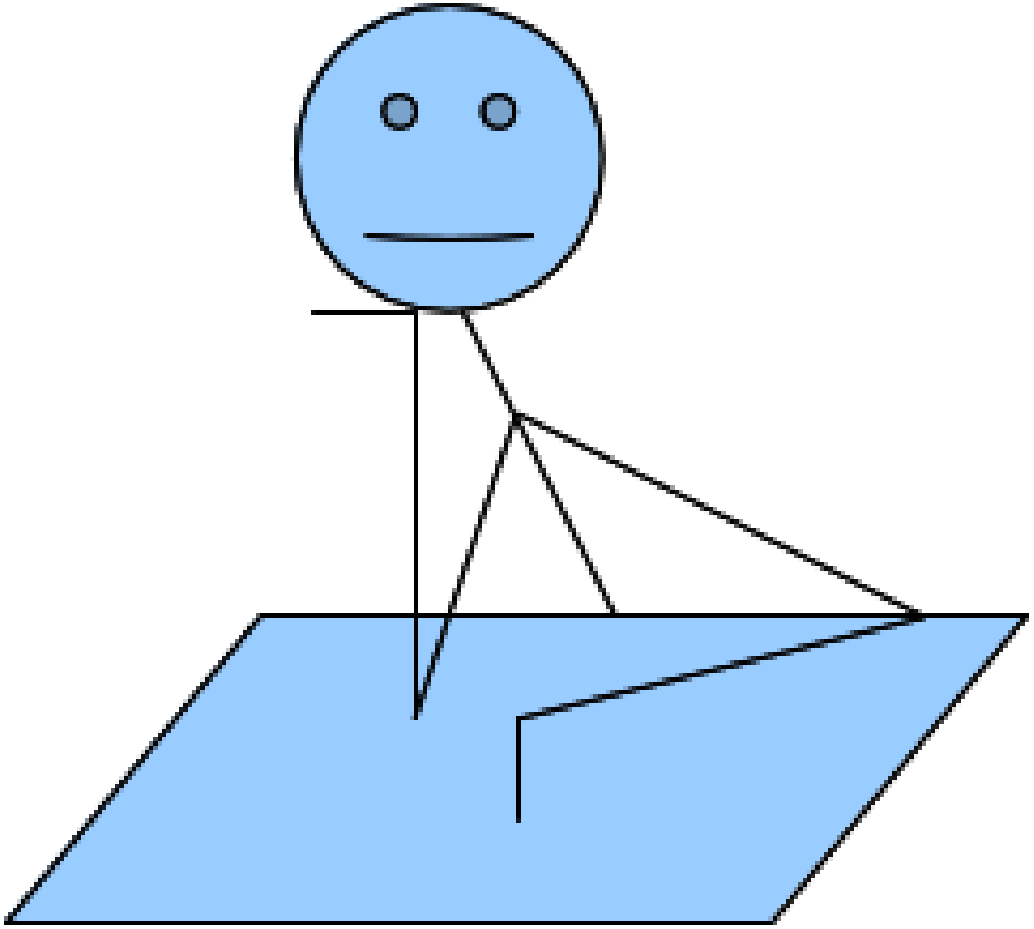
# Example: Which machine am I on?

```
import java.net.*;

public class MyAddress
{
  public static void main (String[] args)
  {
    try
    {
      InetAddress address = InetAddress.getLocalHost();
      String   dottedQuad   = address.getHostAddress();
      System.out.println("Name: " + address);
      System.out.println("IP: " + dottedQuad);
    }
    catch  (UnknownHostException ex)
    {
      System.out.println("Could not find this computer's address.");
    }
  }
}
```

# Subclasses of InetAddress

- Inet4Address
- Inet6Address
  - public boolean isIPv4CompatibleAddress()
    - Returns *`true`* if first 12 bytes are all 0 (has form of IPv4 address)
    - 0:0:0:0:0:0:xxxx:xxxx

# Very nice,
# but how do you do it in C?



# hostname => IP address (C)

- getaddrinfo()
  - Asks DNS to how can contact named machine
  - Returns linked list of how to contact named machine
- freeaddrinfo()
  - *`free()`*s returned linked list
  - Very important, otherwise have memory leak
- Required headers:
  - sys/types.h, sys/socket.h, netdb.h
- Declaration:
  - int getaddrinfo(const char *node, const char *service, const struct addrinfo *hints, struct addrinfo **res);
  - void freeaddrinfo(struct addrinfo *res);
  - const char *gai_strerror(int errcode);

# getaddrinfo() example

```
//  getaddrinfoEx.c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#include <netdb.h>
#include <netinet/in.h>
#include <sys/socket.h>

#ifndef   NI_MAXHOST
#define   NI_MAXHOST 1025
#endif

void    describe       (const char*    nodeNamePtr
                    )
{
  struct addrinfo*      hostPtr;
  struct addrinfo*      run;
  int                   status  = getaddrinfo
                                  (nodeNamePtr,
                                   NULL,
                                   NULL,
                                   &hostPtr);

  if  (status != 0)
  {
    fprintf(stderr,gai_strerror(status));
    return;
  }
```

```
for  (run = hostPtr;  run != NULL;  run = run->ai_next)
{
  struct in_addr      *addr;
  char                hostname[NI_MAXHOST] = "";
  char                ipstr[INET_ADDRSTRLEN];
  int                 error = getnameinfo
                            (run->ai_addr,
                             run->ai_addrlen,
                             hostname,
                             NI_MAXHOST,
                             NULL,
                             0,
                             0);

  if (error != 0)
  {
    fprintf(stderr, "error in getnameinfo: %s\n", gai_strerror(error));
    continue;
  }

  if (*hostname == '\0')
    printf("%-32s:",run->ai_canonname);
  else
    printf("%-32s:", hostname);

  switch  (run->ai_family)
  {
  case AF_INET :    printf("  (IPv4,"); break;
  case AF_INET6 :   printf("  (IPv6,"); break;
  case AF_UNSPEC :  printf("  (IPv4 & IPv6,"); break;
  case AF_UNIX :    printf("  (local Unix,");  break;
  case AF_IPX :     printf("  (Novell,");      break;
  case AF_APPLETALK:printf("  (Appletalk,");  break;
  case AF_PACKET:   printf("  (Lo-level packet,"); break;
  default :         printf("  (Unknown family?,");
  }
```

# getaddrinfo() example, cont'd

```
    switch  (run->ai_socktype)
    {
    case SOCK_STREAM :  printf(" TCP)");    break;
    case SOCK_DGRAM :   printf(" UDP)");    break;
    case SOCK_SEQPACKET:printf(" sequenced, reliable
packet)");  break;
    case SOCK_RAW :     printf(" raw network protocol)");  break;
    case SOCK_RDM :     printf(" reliable w/o ordering)");  break;
    default :           printf(" unknown protocol?)");
    }

    fputc('\n',stdout);

    if  (run->ai_family == AF_INET)
    {
      struct sockaddr_in* ipv4
                        = (struct sockaddr_in*)run->ai_addr;
      addr                    = &(ipv4->sin_addr);
    }
    else
    {
      struct sockaddr_in6* ipv6
                          = (struct sockaddr_in6*)run->ai_addr;
      addr                    = (struct in_addr*)&(ipv6->sin6_addr);
    }
    inet_ntop(run->ai_family,addr,ipstr,sizeof(ipstr));

    if  (ipstr[0] != '\0')
      printf("%s\n\n",ipstr);
    else
      printf("\n");
  }

  freeaddrinfo(hostPtr);
}

int     main            (int argc, char* argv[])
{
  if  (argc < 2)
  {
    fprintf(stderr,"Usage: getAllByName <url>\n");
    exit(EXIT_FAILURE);
  }

  describe(argv[1]);
  return(EXIT_SUCCESS);
}
```

# inet_ntop()

- Is ***supposed*** to turn ip address into a string
- const char* inet_ntop(int af, const void *src, char *dst, socklen_t size)
  - int af: ai_family
  - const void *src:
    - ((struct sockaddr_in*)run->ai_addr)->sin_addr // (run->ai_family == AF_INET)
    - ((struct sockaddr_in6*)run->ai_addr)->sin6_addr // (run->ai_family == /* anything else*/)
  - char* dst: where to write name
  - socklen_t size: length of dst

Your turn!

inet_ntop() may not work for IPv6.

Write your own!

```
struct sockaddr_in6 {
        sa_family_t     sin6_family;   /* AF_INET6 */
        in_port_t       sin6_port;     /* port number */
        uint32_t        sin6_flowinfo; /* IPv6 flow information */
        struct in6_addr sin6_addr;     /* IPv6 address */
        uint32_t        sin6_scope_id; /* Scope ID (new in 2.4)
*/
    };
struct in6_addr {
        unsigned char   s6_addr[16];   /* IPv6 address */
    };
```

# IP address => hostname (C)

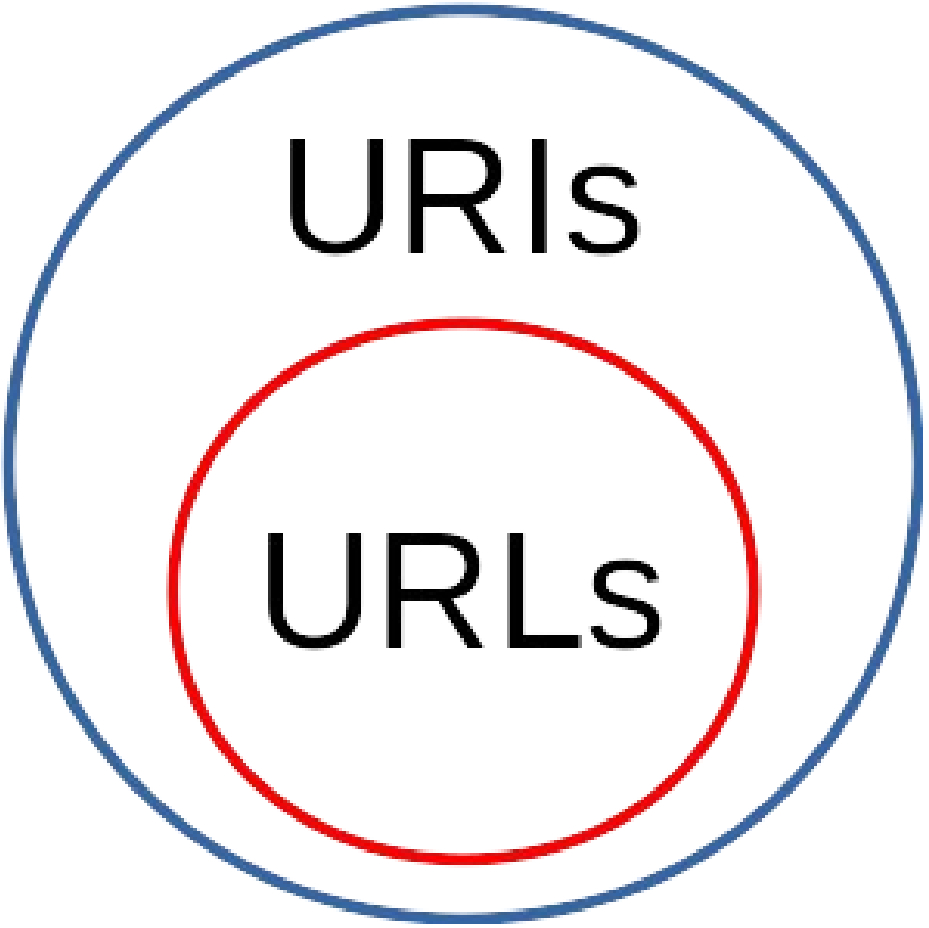
- getnameinfo()
  - Opposite of getaddrinfo()
- Required headers:
  - sys/socket.h, netdb.h
- Declaration
  - int getnameinfo(const struct sockaddr *sa, socklen_t salen, char *host, size_t hostlen, char *serv, size_t servlen, int flags);

# getnameinfo()/inet_pton() example

```
// getnameinfoEx.c
#define _GNU_SOURCE
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <arpa/inet.h>
#include <netdb.h>
int main(int argc, char* argv[]) {
    if  (argc < 2)  {
printf("Give a IP address "
       "(e.g 8.8.8.8 or 216.220.178.116\n"
      );
return(EXIT_SUCCESS);
    }
    struct sockaddr_in sa;
    char node[NI_MAXHOST];

    memset(&sa, 0, sizeof sa);
    sa.sin_family = AF_INET;
    inet_pton(AF_INET, argv[1], &sa.sin_addr);
    /* google-public-dns-a.google.com */

    int res = getnameinfo((struct
sockaddr*)&sa, sizeof(sa),
                          node, sizeof(node),
                          NULL, 0, NI_NAMEREQD);

    if (res) {
        printf("error: %d\n", res);
        printf("%s\n", gai_strerror(res));
    }
    else
        printf("node=%s\n", node);

    return 0;
}
```

# URLs and URIs

- URI
  - Uniform Resource Identifier
  - Can identify a resource by its network location
  - A string representing a URL, email address, new message, etc.
- URL
  - Uniform Resource Location
  - A URI the unambiguously identifies a resource on the Internet
- If you ***want*** things to be found
  - Don’t be shy about creating either,
  - esp. URLs



# URLs

- Syntax:
  - protocol://userInfo@host:port/path?query#fragment
- protocol
  - Chars must be in [a-z,0-9,+,.,-]
- everything else
  - Chars must be in [A-Z,a-z,0-9,-,_,.,!,~]
  - These chars already have a meaning: /?&=
  - Everything else must be escaped with “%” and given in UTF-8
    - http://www.example.com/products%20and%20services.html
- IRI
  - Internationalized Resource Identifier
  - A URI, but whose UTF-8 chars have not been escaped

# URLs

- Syntax:
  - ***protocol***://userInfo@host:port/path?query#fragment
- Supported ***protocols*** include:
  - file, ftp, http, telnet (obvious)
  - mailto (an email address)
  - data (Base64-encoded data included directly in a link)
  - magnet (a resource available for peer-to-peer download)
  - urn (Uniform Resource Name)

# URLs, userInfo

- Syntax:
  - protocol://***userInfo***@host:port/path?query#fragment
- ***userInfo***
  - Form ***username*** (self-evident)
  - Form ***username:password*** :
    - Obviously, do not use unless you want people to know your password
    - Once upon a time you could anonymously login to some ftp servers
      - username: anonymous
      - password: your email address
    - They just wanted to know who was downloading, not trying to stop you

# URLs, host and port

- Syntax:
  - protocol://userInfo@***host***:***port***/path?query#fragment
- host
  - May be ip address (numbers) or hostname
- port
  - Optional: may be implied by the protocol (e.g. “http”) or part of the host (e.g. “www”)
- “authority”
  - userInfo + host + port

# URLs, path

- Syntax:
  - protocol://userInfo@host:port/***path***?query#fragment
- path
  - Unix-style “directory” specification
  - “directory” does not have to correspond to true host directory
    - generally top level is /var/public/html or /srv/public/html
- query
  - Passes additional specifications
  - Common for http servers
- fragment
  - Identifies subpart of document
  - HTML document: <h3 id="xtocid1902914">Comments</h3>
  - Link to fragment: http://www.cafeaulait.org/javafaq.html#xtocid1902914

# Your turn!

The root directory of path
is almost never the
true root directory of host machine

Why not?

# URL Class (Java)

- Factories:
  - public URL (String url) throws MalformedURLException
  - public URL (String protocol, String hostname, String file) throws MalformedURLException
  - public URL (String protocol, String hostname, int port, String file) throws MalformedURLException
  - public URL (URL base, String relative) throws MalformedURLException
- NOTE: Creating a URL instance does not check DNS or reachability!

# Parts of a URL

- public String getProtocol()
- public String getHost()
- public int getPort()
- public int getDefaultPort()
- public String getFile()/public String getPath()
  - Both return full path
  - getFile() also returns query string
  - getPath() does not return query string
- public String getQuery()
  - return *`null`* if no query
- public String getUserInfo()
- public String getAuthority()
  - Returns authority, with or without user info and port

# A program to see which protocols are supported by a particular JVM

```
import  java.net.*;

public class ProtocolTester
{
  public static void main (String[] args)
  {
    if  (args.length < 1)
    {
      System.out.println("Give a URL with a protocol on the
command line, e.g.");
      System.out.println("http://www.wherever.com");
      System.out.println("https://www.wherever.com");
      System.out.println("ftp://www.wherever.com");
      System.out.println("mailto:me@wherever.com");
      System.out.println("telnet://www.wherever.com");
      System.out.println("file:///www.wherever.com");
      System.out.println("gopher://www.wherever.com");
      System.out.println("ldap://www.wherever.com");
      System.out.println("jar://www.wherever.com");
    }
    else
      testProtocol(args[0]);
  }

  private static void testProtocol(String url)
  {
    try
    {
      URL u = new URL(url);

      System.out.println(u.getProtocol() + " is
supported");
    }
    catch (MalformedURLException ex)
    {
      String protocol =
url.substring(0,url.indexOf(':'));

      System.out.println(protocol + " is not
supported");
    }
  }
}
```

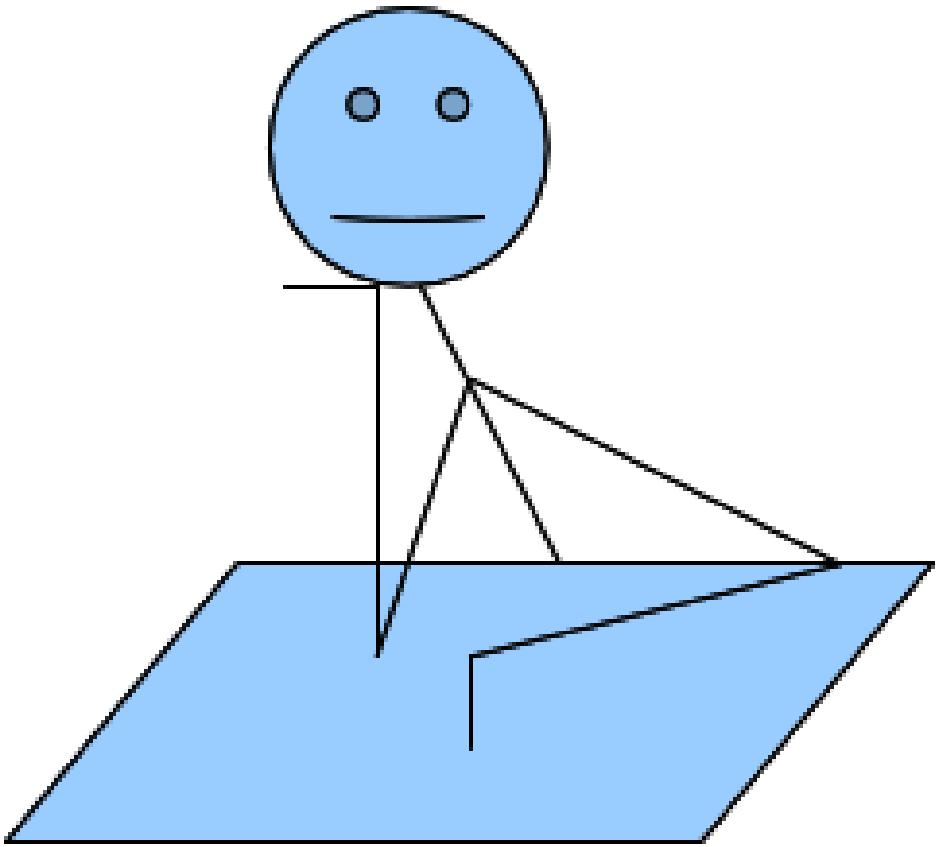
# Watch out!

- URLs are considered equal if-and-only-if these are equal:
  - host, port, path, fragment, query
  - Asks DNS to resolve host
- Thus equals() and hashCode() call DNS
  - equals() on a URL is potentially blocking I/O operation

# Bored student

"*Yeah, yeah, yeah. We can represent URLs,*

*but what I really wants is to access to what they refer.*"



# Easy-Street!

- All we need is one new operation on URLs:
  - URL u = new URL(somePath);
  - InputStream = ***u.openStream()***;

# A program to download a webpage

```
import  java.io.*;
import  java.net.*;

public class    WebPageDownloader
{
  public static void main (String[] args)
  {
    if  (args.length > 0)
    {
      InputStream       in      = null;

      try
      {
        //  Open the URL for reading
        URL u = new URL(args[0]);

        in    = u.openStream();

        //  buffer the input to increase performance
        in    = new BufferedInputStream(in);

        //  chain the InputStream to a Reader
        Reader r = new InputStreamReader(in);
        int    c;

        while  ((c = r.read()) != -1 )
        {
          System.out.print((char) c);
        }
      }
      catch  (MalformedURLException ex)
      {
        System.err.println(args[0] + " is not a
parseable URL");
      }
      catch  (IOException ex)
      {
        System.err.println(ex);
      }
```

# A program to download a webpage

```
      finally
      {
        if  (in != null)
        {
          try
          {
            in.close();
          }
          catch  (IOException e)
          {
            // ignore
          }
        }
      }
    }
  }
}
```

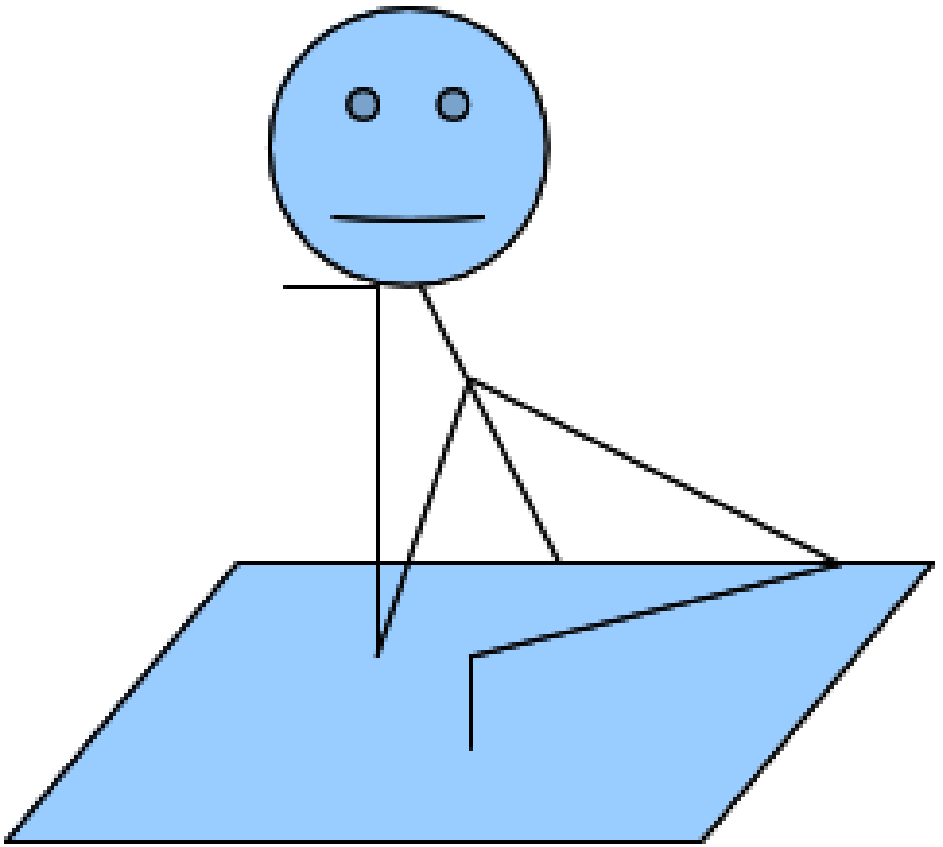
# Your turn!

Write a program to download a given webpage, and then save the images from that webpage.

Assume image filenames appear like:
<img src="img_girl.jpg" alt="Girl in a jacket">

# Smug Java-coding Student

"*Ha ha!  Bet it is not that easy in C/C++*"



# Prof Joe’s retort

“Actually, it is not that bad.

- Several network libraries for C
  - Beast
  - CPT (MIT)
  - ftplib++ (GPL)
  - gnetlibrary (LGPL)
  - GNU Common C++ library’s URLStream class
  - HTTP Fetcher (LGPL)
- Let us look at perhaps the most common one: libcurl

# libcurl

- https://curl.haxx.se/libcurl/
- Starting and stopping:

```
curl_global_init();
// Your code
curl_global_cleanup();
```

- The “full-blown”
  - Multi-threaded, caching
- The “easy”
  - Single-threaded (therefore blocking I/O)
- Link with:
  - gcc …. -lcurl

# libcurl: “easy”

```
curl_global_init(CURL_GLOBAL_ALL); // Turn on
CURL* handle = curl_easy_init(); // start “easy”
curl_easy_setopt(handle, CURLOPT_URL, "http://domain.com/"); // Set URL
curl_easy_setopt(handle, CURLOPT_WRITEFUNCTION, write_data); // Set
writeback
curl_easy_setopt(handle, CURLOPT_WRITEDATA, &data);  // Set arg to writeback
CURLcode status = curl_easy_perform(handle);  // Downloads the page!
curl_easy_cleanup(handle);
curl_global_cleanup();
```

The write-back function:

```
size_t write_data(void *buffer, size_t size, size_t nmemb, void *userp);
```

libcurl delivers as much as possible as often as possible. Your callback function should return the number of bytes it "took care of". If that is not the exact same amount of bytes that was passed to it, libcurl will abort the operation and return with an error code.

# Webpage downloader C (libcurl)

```
// webpageDownloader.c (prev getinmemory.c)
/
***************************************************************
******
 *                                  _  _     ____|_|
 *  Project                     ___| |_| |_  |  _ \| |
 *                             / __| | | | |_) | |
 *                            | (__| |_| |  _ <| |___
 *                             \___|\___/|_| \_\_____|
 *
 * Copyright (C) 1998 - 2018, Daniel Stenberg, <daniel@haxx.se>, et
al.
 *
 * This software is licensed as described in the file COPYING, which
 * you should have received as part of this distribution. The terms
 * are also available at https://curl.haxx.se/docs/copyright.html.
 *
 * You may opt to use, copy, modify, merge, publish, distribute
and/or sell
 * copies of the Software, and permit persons to whom the Software is
 * furnished to do so, under the terms of the COPYING file.
 *
 * This software is distributed on an "AS IS" basis, WITHOUT WARRANTY
OF ANY
 * KIND, either express or implied.
 *

***************************************************************
******/
/* <DESC>
 * Shows how the write callback function can be used to download data
into a
 * chunk of memory instead of storing it in a file.
 * </DESC>
 */

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
```

```
#include <curl/curl.h>

struct MemoryStruct {
  char *memory;
  size_t size;
};

static size_t
WriteMemoryCallback(void *contents, size_t size, size_t nmemb,
void *userp)
{
  size_t realsize = size * nmemb;
  struct MemoryStruct *mem = (struct MemoryStruct *)userp;

  char *ptr = realloc(mem->memory, mem->size + realsize + 1);
  if(ptr == NULL) {
    /* out of memory! */
    printf("not enough memory (realloc returned NULL)\n");
    return 0;
  }

  mem->memory = ptr;
  memcpy(&(mem->memory[mem->size]), contents, realsize);
  mem->size += realsize;
  mem->memory[mem->size] = 0;

  return realsize;
}
```

# Webpage downloader C (libcurl), cont’d

```
int main(int argc, char* argv[])
{
  CURL *curl_handle;
  CURLcode res;
  const char*   nodeNamePtr = (argc<2) ? "https://www.example.com/" : argv[1];

  struct MemoryStruct chunk;

  chunk.memory = malloc(1);  /* will be grown as needed by the realloc above */
  chunk.size = 0;    /* no data at this point */

  curl_global_init(CURL_GLOBAL_ALL);

  /* init the curl session */
  curl_handle = curl_easy_init();

  /* specify URL to get */
  curl_easy_setopt(curl_handle, CURLOPT_URL, nodeNamePtr);

  /* send all data to this function  */
  curl_easy_setopt(curl_handle, CURLOPT_WRITEFUNCTION,
WriteMemoryCallback);

  /* we pass our 'chunk' struct to the callback function */
  curl_easy_setopt(curl_handle, CURLOPT_WRITEDATA, (void *)&chunk);

  /* some servers don't like requests that are made without a user-agent
     field, so we provide one */
  curl_easy_setopt(curl_handle, CURLOPT_USERAGENT, "libcurl-agent/1.0");

  /* get it! */
  res = curl_easy_perform(curl_handle);

  /* check for errors */
  if(res != CURLE_OK) {
    fprintf(stderr, "curl_easy_perform() failed: %s\n",
            curl_easy_strerror(res));
  }
  else {
    /*
     * Now, our chunk.memory points to a memory block that is
chunk.size
     * bytes big and contains the remote file.
     *
     * Do something nice with it!
     */

    printf("%lu bytes retrieved\n", (unsigned long)chunk.size);
    chunk.memory[chunk.size] = '\0';
    printf("%s\n",chunk.memory);
  }

  /* cleanup curl stuff */
  curl_easy_cleanup(curl_handle);

  free(chunk.memory);

  /* we're done with libcurl, so clean it up */
  curl_global_cleanup();

  return 0;
}
```

# Editor’s note . . .

If you strip out the comments of the C program,

then it’s about the same size as the Java one

:)

# Your turn!

Write a program to download a given webpage, and then save the images from that webpage.

Assume image filenames appear like:
<img src="img_girl.jpg" alt="Girl in a jacket">

Start from webPageImageDownloader_inClass.c

# References:


- Elliotte Rusty Harold “*Java Network Programming: 4th Ed.*”
- cboard.cprogramming.com user ueg1990
- cboard.cprogramming.com user algorism
- stackoverflow.com user:126769 (“nos”)