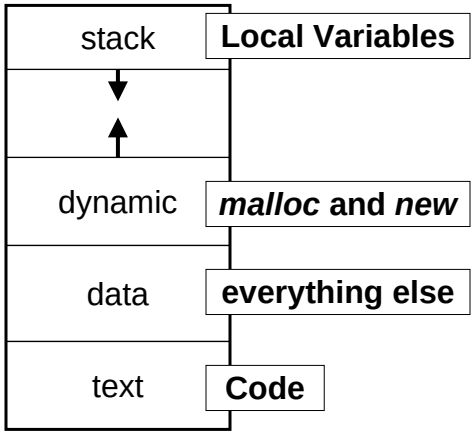
CS-034
It's Magic …

Thomas Doeppner
Pascal Van Hentenryck

Storage



Global Variables

```
int data;

int main() {
  void sub();
  data = 6;
  sub();
  …
  return 0;
}
```

```
void sub() {
  data++;
  printf("data = %d\n",
      data);
  …
}
```


Separate Files

```
/* file x.c */

int data;

int main() {
  void sub();
  data = 6;
  sub();
  …
  return 0;
}
```

```
/* file y.c */

extern int data;

void sub() {
  data++;
  printf("data = %d\n",
      data);
  …
}
```

gcc –o prog x.c y.c

Really Separate Files

```
/* file x.c */

static int data = 6;

int main() {
  void sub();
  sub();
  printf("data = %d\n",
      data);
  …
  return 0;
}
```

```
/* file y.c */

static int data = 3;

void sub() {
  data++;
  printf("data = %d\n",
      data);
  …
}
```


Static Local Variables

```
int *sub1() {
  int var;
  …
  return &var;
  /* amazingly illegal */
}
```

```
int *sub2() {
  static int var;
  …
  return &var;
  /* (amazingly) legal */
}
```

Static Member Variables

```
class X {
public:
  X(int v) : var1(v) {}
  int val() {
      return var1 + var2;
  }
private:
  int var1;
  static int var2;
};

int X::var2 = 7;
```

```
int main() {
  X x(2);
  cout << x.val() << endl;
  return 0;
}
```

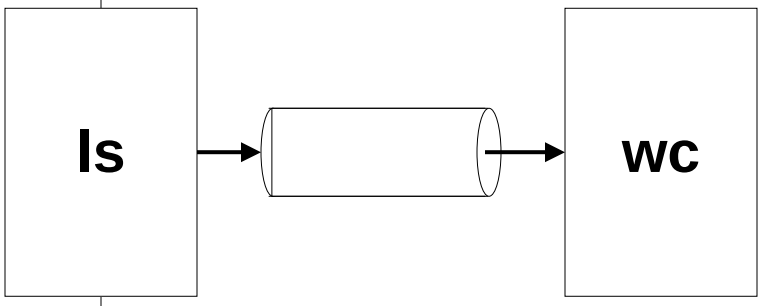

Static Member Functions

```
class X {
public:
  X(int v) : var1(v) {}
  int val() {
      return var1 + var2;
  }
   static int get() {
     return var2;
   }
private:
  int var1;
  static int var2;
};
int X::var2 = 7;
```

```
int main() {
   cout << X::val();

  X x(2);
  cout << x.val() << endl;
  return 0;
}
```

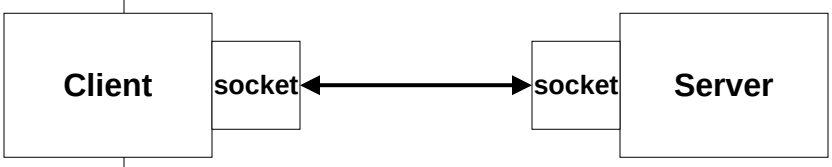
Pipes



Pipe Code

```
int main( ) {
  int fd[2];
  pipe(fd);
  if (fork() == 0) {
    dup2(fd[0], 0);
    close(fd[0]);
    close(fd[1]);
    execlp("wc", "wc",
        "-l", 0);
  }
  if (fork() == 0) {
    dup2(fd[1], 0);
    close(fd[0]);
    close(fd[1]);
    execlp("ls", "ls",
        "-1", 0)
  }
  close(fd[0]);
  close(fd[1]);
  while (wait(0) >= 0)
    ;
}
```

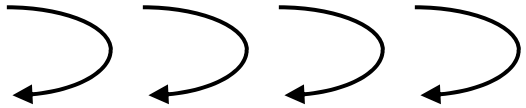
Communicating Over the Internet



Steps …

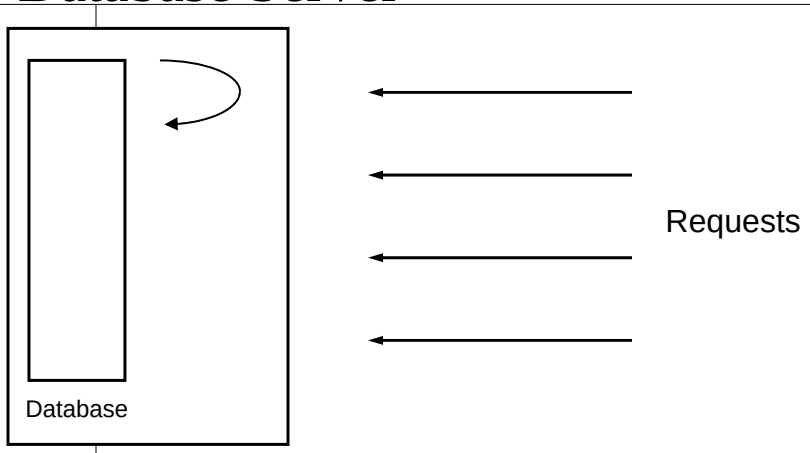
- **Client**
  - **create socket**
    - `socket`
  - **connect to server**
    - `connect`
  - **send message**
    - `write`
- **Server**
  - **create socket**
    - `socket`
  - **set address**
    - `bind`
  - **wait for connection**
    - `listen`
  - **get connection**
    - `accept`
  - **get message**
    - `read`

Multithreaded Programming

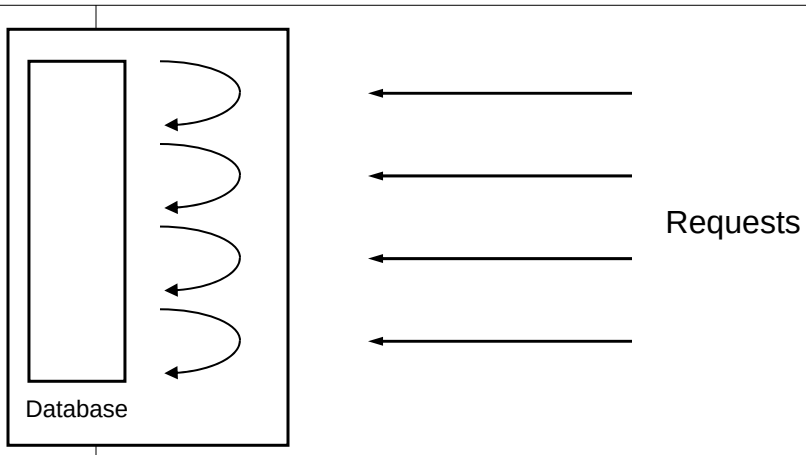


- **Why?**
  - **many things are easier to do with threads**
  - **some things run faster with threads**

Single-Threaded Database Server



Multithreaded Database Server



Creating a Thread

```
start_servers( ) {
    pthread_t thread;
    int i;
    for (i=0; i<nr_of_server_threads; i++)
        pthread_create(&thread,    // thread ID
            0,                     // default attributes
            server,                // start routine
            argument);             // argument
}

void *server(void *arg) {
    while(1) {
        /* get and handle request */
    }
}
```

Example (1)

```
#include <stdio.h>
#include <pthread.h>

#define M  3
#define N  4
#define P  5

int A[M][N];
int B[N][P];
int C[M][P];

void *matmult(void *);
```

```
main( ) {
   int i;
   pthread_t thr[M];
   int error;

   /* initialize the matrices
     ... */

   ...
```


Example (2)

```
   for (i=0; i<M; i++) {  // create the worker threads
      pthread_create(&thr[i], 0, matmult, (void *)i));
   }
   for (i=0; i<M; i++)     // wait for workers to finish
     pthread_join(thr[i], 0)

   /* print the results ... */
}
```

Example (3)

```
void *matmult(void *arg) {
   int row = (int)arg;
   int col;
   int i;
   int t;

   for (col=0; col < P; col++) {
      t = 0;
      for (i=0; i<N; i++)
         t += A[row][i] * B[i][col];
      C[row][col] = t;
   }
   return(0);
}
```


Compiling It

```
Linux% gcc –o mat mat.c –D_REENTRANT \
     -lpthread
```

Mutual Exclusion

Threads and Mutual Exclusion

Thread 1:

```
x = x+1;
 /*
   ld   r1,x
   add  r1,1
   st   r1,x
 */
```

Thread 2:

```
x = x+1;
 /*
   ld   r1,x
   add  r1,1
   st   r1,x
 */
```

Threads and Synchronization

```
pthread_mutex_t m =
     PTHREAD_MUTEX_INITIALIZER;
     // shared by both threads
int x;// ditto

  pthread_mutex_lock(&m);

  x = x+1;

  pthread_mutex_unlock(&m);
```


Libraries

- **Collections of useful stuff**
- **Incorporate items into your program**
- **Replace existing items with new stuff**
- **Often ugly …**

Creating a Library

```
% gcc –c sub1.c sub2.c sub3.c
% ls
sub1.c          sub2.c          sub3.c
sub1.o          sub2.o          sub3.o
% ar cr libpriv1.a sub1.o sub2.o sub3.o
% ar t libpriv1.a
sub1.o
sub2.o
sub3.o
%
```


Using a Library

```
% cat prog.c
int main() {
  sub1();
  sub2();
  sub3();
}
% cat sub1.c
void sub1() {
  puts("sub1");
}
```

```
% gcc –o prog prog.c –L. –lpriv1
```

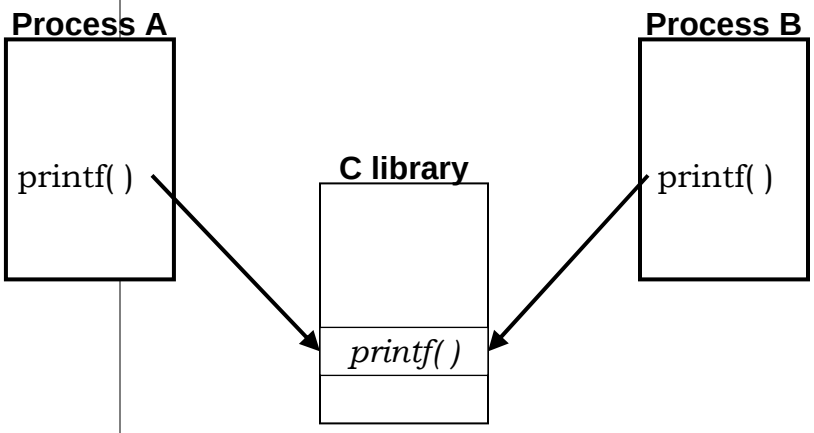
Where does *puts* come from?

```
%gcc –o prog prog.c –L. \
  -lpriv1 –L/lib -lc
```

Substitution

```
% ar t libmystdio.a
puts.o
% gcc –o prog prog.c –L. –lpriv1 –lmystdio
%
```

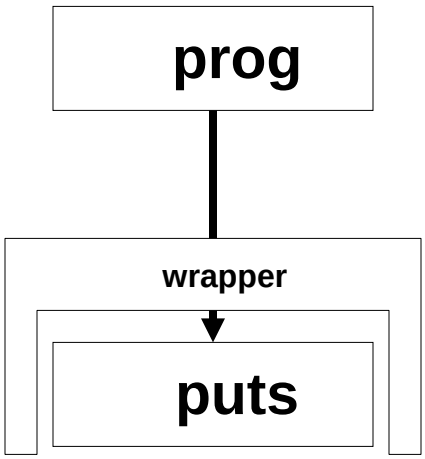

Shared Libraries



Creating a Shared Library

```
% gcc –fpic –c libstdio.c
% ld –shared –soname libstdio.so.1 \
    –o libstdio.so.1
% gcc –o prog prog.c –L. –lpriv1 –lmystdio
```


Interpositioning



How To?

```
#include <dlfcn.h>

int puts(const char *s) {
  int (*pptr)(const char *);

  pptr = (int(*)())dlsym(RTLD_NEXT, "puts");

  write(2, "calling myputs: ", 16);
  return (*pptr)(s);
}
```


Compiling/Linking It

```
% gcc –fPIC –c myputs.c –D_GNU_SOURCE
% ld –shared –soname libmyputs.so.1 \
  -o libmyputs.so.1 myputs.o –ldl
% gcc -o tputs tputs.c ./libmyputs.so.1 \
  -Wl,-rpath .
```

What's Going On …

- **gcc/ld**
  - **compiles code**
  - **does static linking**
    - **searches list of libraries**
    - **adds references to shared objects**
- **runtime**
  - **program invokes *ld.so* to finish linking**
    - **searches list of shared objects**
  - ***dlsym* invokes *ld.so* to do more linking**

Delayed Wrapping

- **LD_PRELOAD**
  - **environment variable checked by *ld.so***
  - **specifies additional shared objects to search (first) when program is started**

Example

```
% gcc –o tputs tputs.c
% ./tputs
This is a boring message.
% setenv LD_PRELOAD ./libmyputs.so.1
% ./tputs
calling myputs: This is a boring message.
%
```


Can We Do It With Printf?

- **Takes variable number of arguments**
- **See “man stdarg” for details on handling such programs**

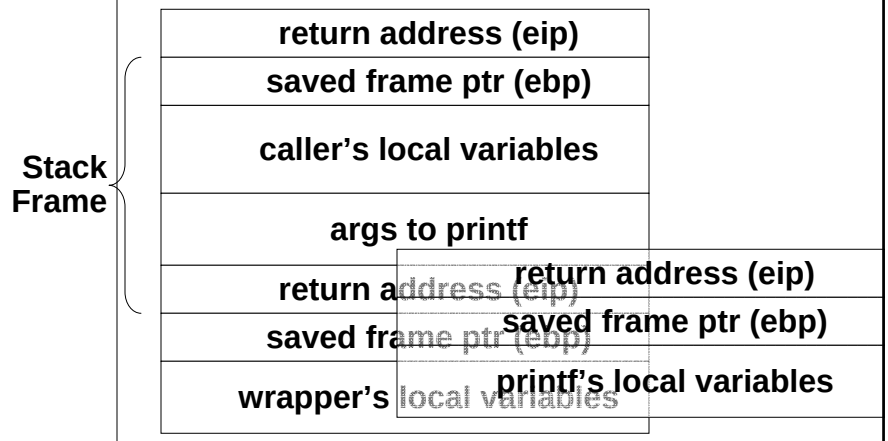
```
#include <dlfcn.h>

int printf(const char *s, …) {
  int (*pfptr)(const char *, …);

  pfptr = (int(*)())dlsym(RTLD_NEXT, "printf");

  write(2, "calling myprintf: ", 18);
  return (*pfptr)(s, ???????);
}
```

The Runtime Stack



Inline Assembler Code

```
#include <dlfcn.h>

int printf(const char *s, ...) {
  int (*pfptr)();

  pfptr = (int(*)())dlsym(RTLD_NEXT, "printf");
  write(2, "printf: ", 8);
  asm("movl -4(%ebp), %eax");
         /* save pfptr address */
  asm("leave");         /* pop off stack frame */
  asm("jmp *%eax");     /* jump to real printf */
  /* (*pfptr)(); */
  return  0;
}
```

Enough of Ugly …