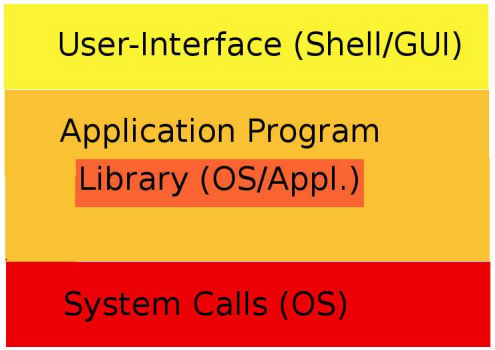
Library: Static Linking (1)

library functions incorporated into program (at compile time)



Library: Static Linking (2)

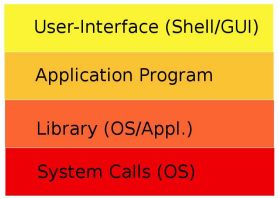
+ applications are more robust (run *standalone*)
+ cases saving performance: many calls to library routines
+ errors in new library version limited to corresp. programs
- program size bigger
- changes in library require recompiling program

static linking is enabled by

```
gcc -static ...
```


Dynamic Linking

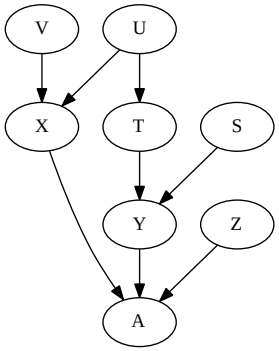
library functions load on demand into program (default, at run-time)



+ saves main memory, smaller size of executable
+ cases saving performance: already in mem, less likely paged out
+ simplifies security patches
- errors are devastating
- compatibility with library has to be ensured $\rightsquigarrow$*dependency hell*

The Dependency Hell (with Dynamic Linking)

also called DLL-hell in Windows, JAR-hell in Java,...



$\rightsquigarrow$package manager

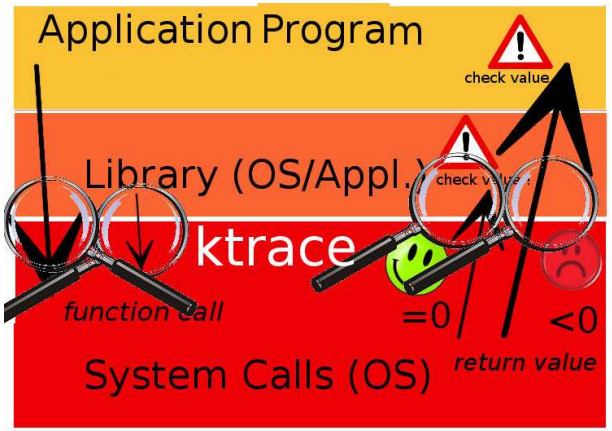
Case Study: A C–Program


```
#include <stdio.h>

int main(int argc, char **argv)
{
        FILE *f;

        if (argc<=1) return 1;
        f=fopen(argv[1],"w");
        if (!f) return 1;
        fprintf(f,"hello world\n");
        fclose(f);

        return 0;
}
```


Case Study: Which Shared Libraries Are Involved?

```
$ ldd filetest
filetest:
        libc.so.6 => /lib/libc.so.6 (0x2807c000)
```


Watching this Call: ktrace (BSD)



(Linux users use `strace`)

ktrace / kdump

```
ktrace ./testprog myfile
kdump -tcn | less
...
 16114 testprog CALL  open(0xbfbfe8f7,0x601,0x1b6)
 16114 testprog NAMI  "myfile"
 16114 testprog RET   open 3
...
 16114 testprog CALL  write(0x3,0x804b000,0xc)
 16114 testprog RET   write 12/0xc
 16114 testprog CALL  close(0x3)
 16114 testprog RET   close 0
 16114 testprog CALL  exit(0)
```


Examples of System Calls

Files	Creating a Channel	creat()
		open()
		close()
	Input/Output	read()
		write()
	Random Access	lseek()
	Channel Duplication	dup()
	Aliasing and Removing	link()
	Files	unlink()
	File Status	stat()
	Access Control	access()
		chmod()
		chown()
		umask()
	Device Control	ioctl()

Processes	Process Creation	exec()
		fork()
	Termination	wait()
		exit()
	Process Owner and Group	getuid()
		geteuid()
		getgid()
		getegid()
	Process Identity	getpid()
		getppid()
	Process Control	signal()
		kill()
		alarm()
	Change Working Directory	chdir()

IPC	Pipelines	pipe()
	Messages	msgget()
		msgsnd()
		msgrcv()
		msgctl()
	Semaphores	semget()
		semop()
	Shared Memory	shmget()
		shmat()
		shmdt()

C–Program with System Calls


```
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <unistd.h>
#include <stdio.h>

int main(int argc, char **argv)
{
        int fd; /* file descriptor */
        if (argc<=1) return 1;
        fd=open(argv[1],O_CREAT|O_WRONLY,S_IRUSR|S_IWUSR);
        if (fd<0)
        {
                perror("open");
                return 1;
        }
        if (write(fd,"hello world\n",12)<0)
        {
                perror("write()");
                return 1;
```



```
        }
        if (close(fd)<0)
        {
                perror("close()");
                return 1;
        }

        return 0;
}
```

Commands/Functions/Kernel: The uname Command (1)

Which kernel name?

```
isl-c-01:~$ uname -s
FreeBSD
```

Which hostname?

```
isl-c-01:~$ uname -n
isl-c-01
```

Which operating system release?

```
isl-c-01:~$ uname -r
9.0-STABLE
```


The uname Command (2)

Which kernel version?

```
isl-c-01:~$ uname -v
FreeBSD 9.0-STABLE #0: Fri Jan 27 20:10:29 CET 2012
root@isl-c-01:/usr/obj/usr/src/sys/ISL-L-01
```

Which machine hardware?

```
isl-l-01:~$ uname -m
amd64
```

All of the uname information:

```
isl-c-01:~$ uname -a
FreeBSD isl-c-01.htw-saarland.de
9.0-STABLE FreeBSD 9.0-STABLE
#0: Fri Jan 27 20:10:29 CET 2012
root@isl-l-01:/usr/obj/usr/src/sys/ISL-L-01
amd64
```


UNIX is Written in C

. . . so there is a uname() C function . . .


```
UNAME(3)      FreeBSD Library Functions Manual       UNAME(3)

NAME
     uname -- get system identification

LIBRARY
     Standard C Library (libc, -lc)

SYNOPSIS
     #include <sys/utsname.h>

     int
     uname(struct utsname *name);

DESCRIPTION
     The uname() function stores NUL-terminated strings of
     information identifying the current system into the structure
     referenced by name.

     The utsname structure is defined in the <sys/utsname.h> header file,
```

```
     and contains the following members:

           sysname       Name of the operating system implementation.

           nodename      Network name of this machine.

           release       Release level of the operating system.

           version       Version level of the operating system.

           machine       Machine hardware platform.

RETURN VALUES
     The uname() function returns the value 0 if successful; otherwise
     the value -1 is returned and the global variable errno is set to
     indicate the error.
```


struct utsname

from /usr/include/sys/utsname.h

```
#ifndef SYS_NMLN
#define SYS_NMLN 256  /* User can override. */
#endif

struct utsname {
   char    sysname[SYS_NMLN];   /* Name of this OS. */
   char    nodename[SYS_NMLN];  /* Name of network node. */
   char    release[SYS_NMLN];   /* Release level. */
   char    version[SYS_NMLN];   /* Version level. */
   char    machine[SYS_NMLN];   /* Hardware type. */
};
```


A uname C Program


```
#include <sys/utsname.h>
#include <stdio.h>
int main()
{
        struct utsname buffer;
        if (uname(&buffer)==0)
        {
                puts("system information:");
                printf("sysname: %s\n",buffer.sysname);
                printf("nodename: %s\n",buffer.nodename);
                printf("release: %s\n",buffer.release);
                printf("version: %s\n",buffer.version);
                printf("machine: %s\n",buffer.machine);
        }
        else
                perror("uname");
        return 0;
}
```

ktrace of uname C Program

```
 60550 ktrace   CALL  execve(0xbfbfe367,0xbfbfe22c,0xbfbfe234)
 60550 ktrace   NAMI  "./uname-read"
 60550 ktrace   NAMI  "/libexec/ld-elf.so.1"
 60550 uname-read RET   execve 0
...
 60550 uname-read CALL  open(0x28073040,0,0)
 60550 uname-read NAMI  "/lib/libc.so.6"
 60550 uname-read RET   open 3
...
 60550 uname-read CALL  __sysctl(0xbfbfdc80,0x2,0xbfbfdce0,0xbfbfdc7c,0,0)
 60550 uname-read RET   __sysctl 0
...
 60550 uname-read CALL  exit(0)
```


sysctl

Management Information Base (sort of Registry of a UNIX System)

for kernel state

```
$ sysctl kern.ostype
kern.ostype: FreeBSD

$ sysctl kern.osrelease
kern.osrelease: 6.3-RELEASE-p2

$ sysctl hw.model
hw.model: Intel(R) Pentium(R) 4 CPU 1.80GHz
```