PURDUE
UNIVERSITY®

CS 25200: Systems Programming

Lecture 5: File System Hierarchy, Program Structure, Generation, and Loading

Prof. Turkstra

Lecture 05

- UNIX file system structure
- Processes and users
- Program segments
- Executable formats
- Building a program
- Libraries
- Loader

UNIX File System

- *NIX has a hierarchical file system
- Root of entire tree is denoted with /
- Disks and network shares can be mounted anywhere inside the hierarchy

Hierarchy

- /bin – historically contained fundamental utilities (ls, cp, etc)
  - Often now a symlink to /usr/bin
- /usr - “user file system”
  - Used to be split off on separate storage, usually not anymore
- /boot – files necessary for startup (e.g., initial kernel image)

- /dev – peripheral and other devices
- /tmp – temporary files that do not survive a reboot
- /var – files that may change frequently (variable)
  - Email, logs, databases, etc
- /sbin - “system (superuser) binaries”
  - Utilities needed to start and maintain/recover the system
  - Also symlinked to /usr/sbin

- /etc – system-wide configuration files
  - Init system scripts
    - /etc/rc.d
    - /etc/rc.*
  - User and group information
    - /etc/passwd, /etc/groups
- /home – user home directories
- /lib – essential libraries
  - Often split into lib, lib64 now
  - Also usually symlinked into /usr/lib*

- /usr/include – header files for libraries and kernel
- /proc – virtual file system that provides information about processes through files
- /sys – other hardware and kernel-related information

Processes

- top, ps (-e, -ax, -f, -u)

Users

- UNIX was designed as a multiuser system
- Database of users exists in /etc/passwd
  - ...but not for Purdue machines (NIS/LDAP)
  - Each user has a unique uid
  - Passwords are now stored in /etc/shadow

- useradd – create a new user
- passwd – change a user’s password
- There exists a special user called “root”
  - Root can do anything
    - Modify files, change permissions/ownership, mount/umount, rm anything, execute anything, etc etc
  - To become root, you use the “su” command. (super user)

Got root?

- Many times systems are configured to use "sudo". Sudo allows regular users to run certain (specified) things as root. See /etc/sudoers
  - Does not require the root user's password, unlike su
  - Can "sudo su" to get a root shell, if you have the right privileges
- Root shells are dangerous
  - `$ rm -rf / home/turkstra/somefile`

Groups

- A group is a collection of users
- Users may belong to multiple groups
  - "Supplemental groups"
  - usermod -S, groupadd, groupmod
  - /etc/group

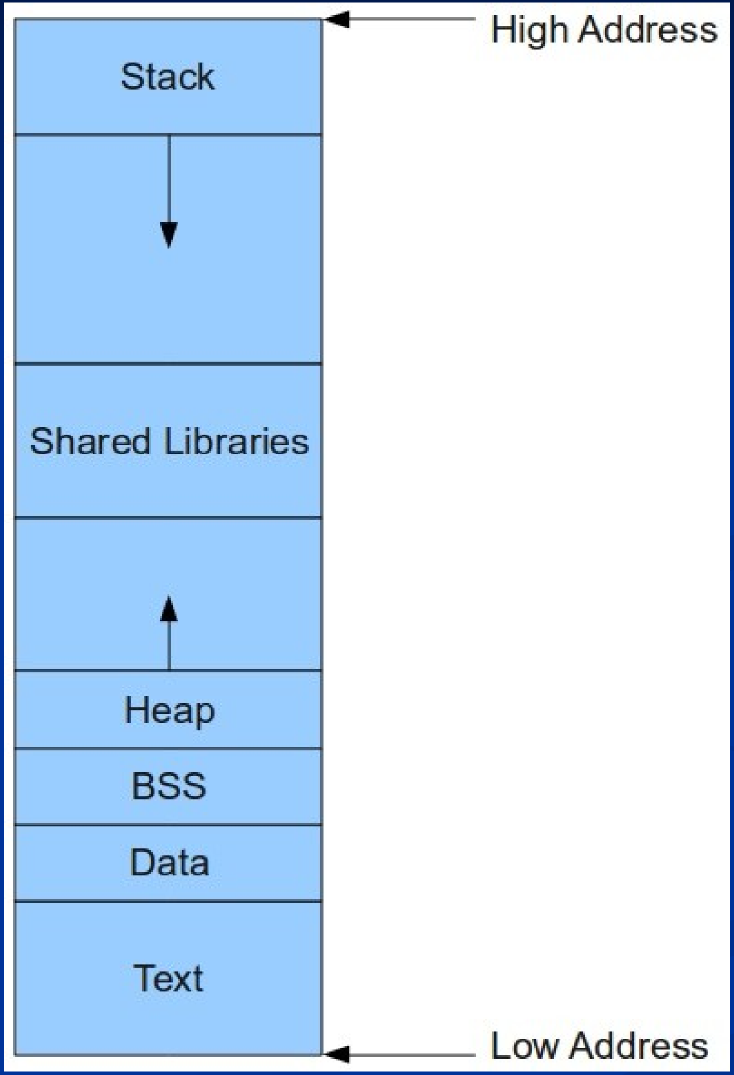
Network Information Service (NIS)

- Formerly Yellow Pages or YP
- Client-server directory service protocol for distributing information like users and groups
- See “ypcat”

Program vs. process

- A program is an executable file that contains a set of instructions
  - Usually stored on disk or other secondary storage
- A process is a program in execution
  - It resides, at least partially, in memory

Process memory layout



32-bit vs 64-bit

- 32-bit systems usually have shared libraries at the lowest address, followed by the text segment
- Starting addresses differ
  - Text or code usually starts 0x400000 on 64-bit, 0x8047000 on 32-bit

Text segment

- Also called the code segment
- Contains actual program instructions and any statically linked libraries
- Often read only and executable
  - Self-modifying code

Demo

- hello.c

```
#include <unistd.h>
int main(int argc, char *argv[])
{
  int ret = 0xbeefbeef;
  ret = write(1, "Hello\n", 6);
  return 0;
}
```

```
$ gcc -S hello.c
$ objdump -Dl a.out
```

Data segment

- Initialized global variables and static strings

BSS

- Block started by symbol
- Holds uninitialized global variables
  - By C convention are initialized automatically to 0

Heap

- Dynamically allocated memory
  - i.e., obtained via `malloc()`
- Grows upward as memory is requested
  - Upward → increasing addresses

Stack

- Holds local, temporary, or automatic, variables
- Arguments passed during a function call
- Information needed to return to a previous point in the program
- Grows downward (decreasing addresses)

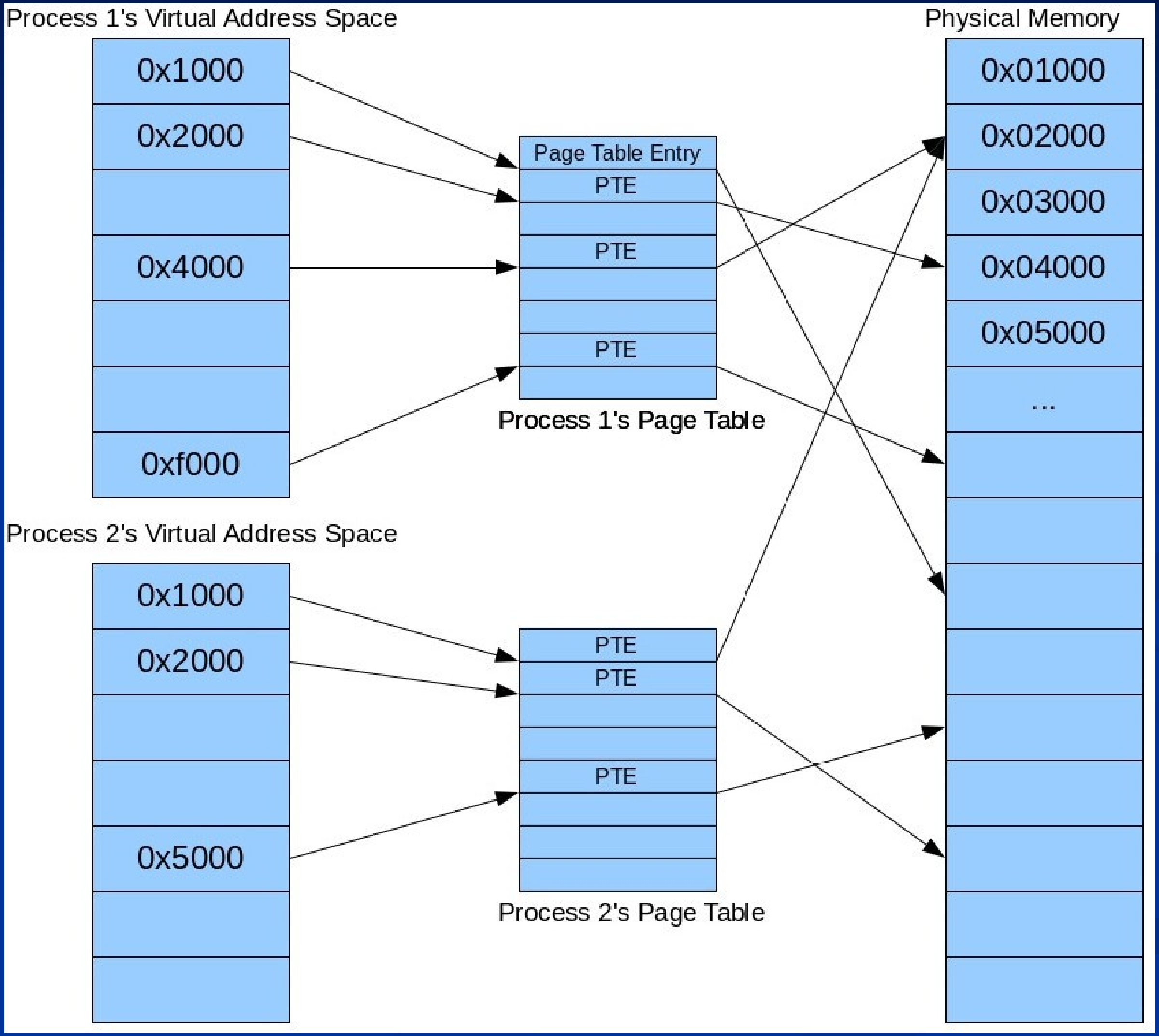
Stack

- A stack is a very common data structure used in programs and architectures
- Stacks are generally LIFO queues
  - Last in, first out
- Two operations
  - PUSH – add something to the stack
  - POP – retrieve the most recent item

Shared libraries

- Shared libraries permit the same text segment to be used in multiple processes
  - Through the magic of virtual memory
  - Have their own data and BSS segments
    - Usually copy-on-write

Virtual memory



Demo

- $ cat /proc/self/maps

- howdy.c:

```
int a = 5;                // Stored in data
int b[20];                // Stored in BSS
int main() {              // Stored in text
  int x;                  // Stored in stack
  int *p = (int *) malloc(sizeof(int)); // Heap
  printf("%p\n%p\n%p\n%p\n%p\n", &a, &b, &x, p,
         main);
}
```

Gaps

- Often gaps between segments
- Attempting to access an address in an unmapped region causes the OS to send the process a signal: SIGSEGV
  - Segmentation violation
  - By default program is immediately terminated and dumps core
- This also happens if you access a mapped but protected region
  - E.g., try executing in the data segment

Core dump file

- $ man 5 core
- File containing an image of the process' memory at time of termination
- Can be used with gdb (or other debuggers)
- May have to enable it (e.g., on data.cs.purdue.edu)
  - $ ulimit -c unlimited

Program

- File in a particular format containing necessary information to load an application into memory and execute it
  - Often time part of this is split off into the "loader" and libraries
- Programs include:
  - Machine instructions
  - Initialized data
  - List of library dependencies
  - List of memory sections
  - List of values determined at load time

Executable file formats

- Fully "compiled" programs can come in many different formats…
  - ELF – Executable Linux File
    - Used by most recent UNIX systems (e.g., Solaris, Linux)
  - PE – Portable Executable
    - Used by Windoze
  - Mach-O – Mach object [file format]
    - Used by macOS/iOS
  - COFF – Common Object File Format
    - Also windoze, some embedded systems - historically System V Unix
  - a.out – Used in BSD (Berkeley Standard Distribution)
    - Restrictive, rarely seen anymore

ELF

- File header
  - Magic number
  - Version
  - Target ABI
  - ISA
  - Entry point
  - Pointers to
    - Program header
    - Section header
  - etc

Program header

- How to create the process image
  - Segments
  - Types
  - Flags
  - File offset
  - Virtual address
  - Size in file
  - Size in memory

Section header

- Type (data, string, notes, etc)
- Flags (writable, executable, etc)
- Virtual address
- Offset in file image
- Size
- Alignment

- readelf --headers /bin/ls
- objdump -p, -h, -t

Building a program

- Start with source code
  - abs.c
- Prepocessor
- Compiler
- Assembler
- Linker

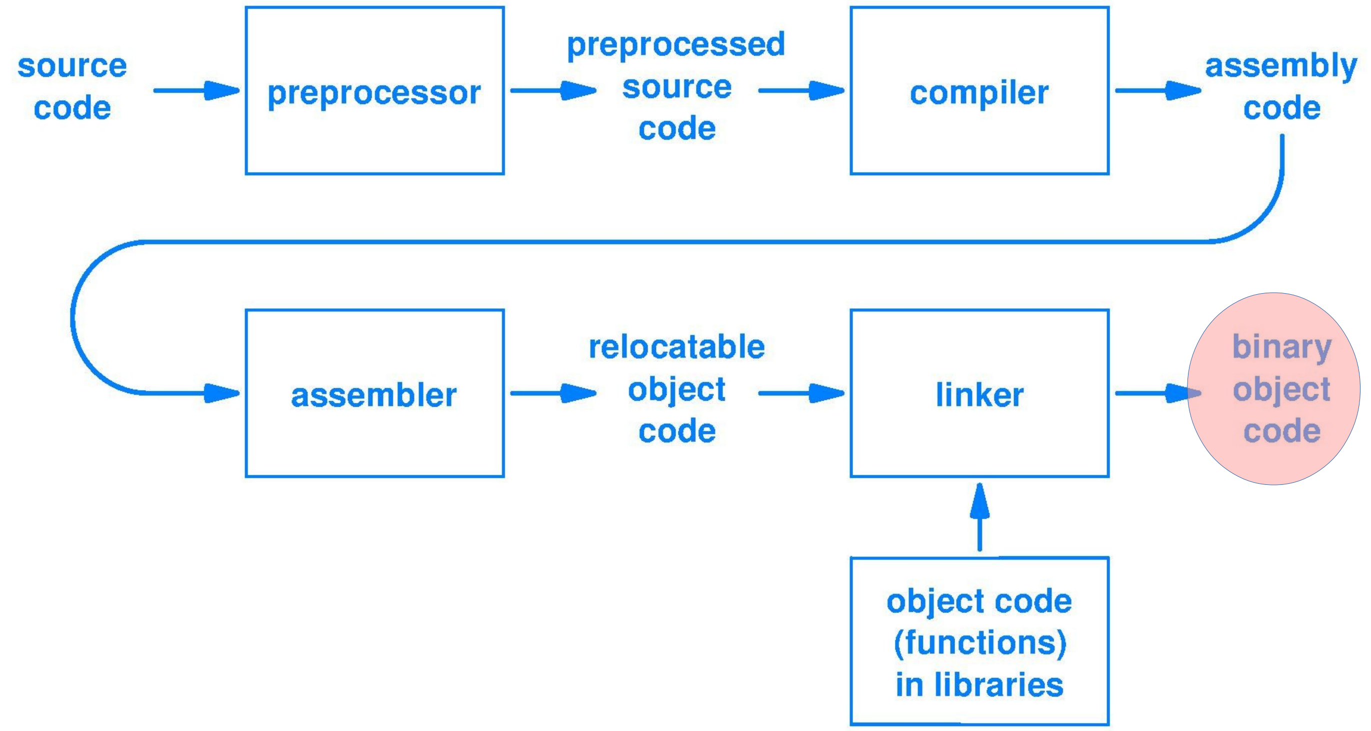


Figure 4.6 The steps used to translate a source program to the binary object code representation used by a processor.

Preprocessor

- When a .c file is compiled, it is first scanned and modified by the preprocessor before being handed to the real compiler
- Finds lines beginning with #, hides them from the compiler, or takes some action
- #include, #define
- #ifdef, #else, #endif

Why macros?

- Run time efficiency
  - No function call overhead
- Passed arguments can be any type
  - #define MAX(x,y) ((x) > (y) ? (x) : (y))
  - Works with ints, floats, doubles, even chars

- Can do math
  - #if (FLAG % 4 == 0) || (FLAG == 13)
- Macros
  - #define INC(x) x+1
  - No semi-colon
  - Have to be careful
    - #define ABS(x) x < 0 ? -x : x
    - ABS(B+C)

- Parentheses around substitution variables

```
#define ABS(x)  ( (x) < 0 ? -(x) : (x) )
```

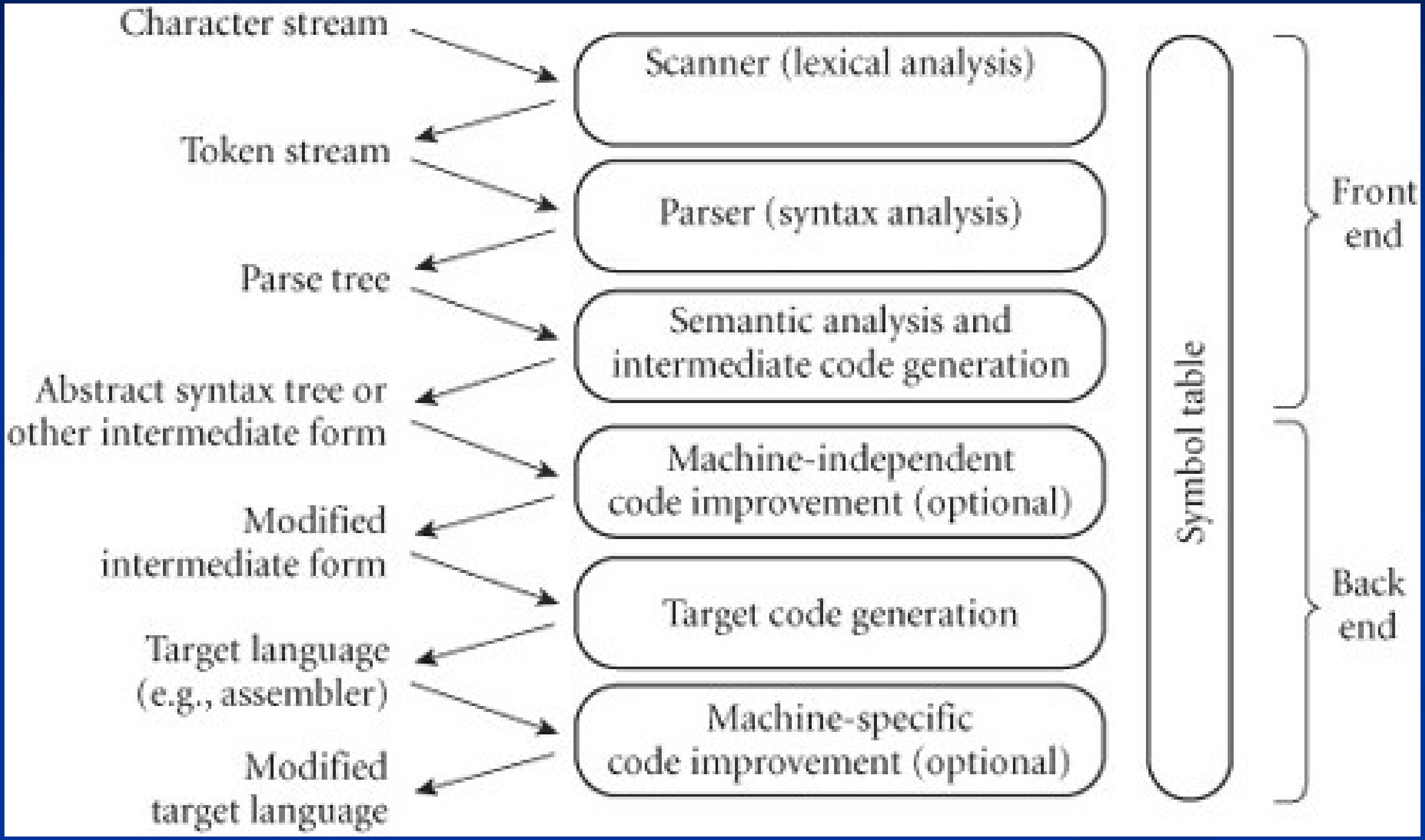
Lots of other tricks

```
printf("The date is %s\n", __DATE__);
```

- Most preprocessor features are used for large/advanced software development practices

- gcc -E file.c > output.i

Compiler?


Character stream
Token stream
Parse tree
Abstract syntax tree or other intermediate form
Modified intermediate form
Target language (e.g., assembler)
Modified target language
Scanner (lexical analysis)
Parser (syntax analysis)
Semantic analysis and intermediate code generation
Machine-independent code improvement (optional)
Target code generation
Machine-specific code improvement (optional)
Symbol table
Front end
Back end

* http://www.cs.montana.edu/~david.watson5/

- gcc -S

Assembler

- Discussed in CS 250 – Computer Architecture
- gcc -c

  nm -v
- Really uses as

Libraries

- Libraries are just collections of object files
  - Internal symbols are indexed for fast lookup by the linker
- Searched for symbols that aren't defined in the program
  - Symbol found, pull it into executable (static)
  - Otherwise include a pointer to the file, loaded by loader

Statically linked

- Faster, to a degree
- Portable
- Larger binaries
- Fixed version, no updates
- File extension .a

Dynamically linked

- More complexity
- Easy to upgrade libraries
  - Vulnerabilities
- Have to manage versions
- Loader re-links every time program is executed

```
readelf --dynamic /bin/ls
ldd /bin/ls
```

End result

- gcc -o abs abs.c
  nm -v ./abs

Loader

- Essential step in starting a program
- Historically allocated space for all sections of the executable (text, data, bss, etc)
- Now simply establishes mappings
  - Page faults actually populate the memory
  - For executable as well as (shared) libraries

- Also resolves any values in the executable to point to the functions/variables in the shared libraries
- Jumps to _start
  - init()'s all libraries
  - _then calls main()
  - ...and exit()
- Sometimes loaders are called "runtime linkers"

Interpreter

```
readelf --headers /bin/ls
```

Lazy binding

- Binding a function call to a library can be expensive
  - Have to go through code and replace the symbol with its address
- Delay until the call actually takes place
  - Calls stub PLT function
  - Invokes dynamic linker to load the function into memory and obtain real address
    - Rewrites address that the sub code references
    - Only happens once
- Procedure Lookup Table (PLT)

Questions?