Address Space

text=instructions

data=initialized global variables

BSS=uninitialized global variables

heap-malloc, new, dynamic memory allocation

Shared Lib (Dynamic libraries)- has their own text, data, BSS

stack-local variables, grows downward

&a is the reference operator - “address of variable a”.

\*p is the dereference operator - “give content of the address pointed by p”.

int a = 10; // say address of 'a' is 2000;

int \*p = &a; // p is pointing to the address of a

int c = \*p; //give the content of the address pointed by 'p', which is a = 10

executable file format

ELF- Executable link file-used in most UNIX system, Linux, Solaris

COFF- Common object file format- used in windows system

a.out – used in BSD, Berkeley standard distribution

preprocessor: expands #define, #include, #ifdef etc & generate hello.i

compiler: compiles and generates an assembly instruction hello.s

assembler: assembles hello.s and generates object file hello.o (value, size, type, name, etc)

linker: puts together object files(.o) & static libs(.a) & shared libs (.so) ->hello.c

loader-runtime linker: allocate space for memory sections, loads exe and shared libs into memory

chmod 400 readable, 500 runable, 600 writable

set p=NULL after a free()

implement 2>

error check when open file, pipe, etc

The minimum size of an allocated chunk is 32 bytes. This is because when that chunk becomes unallocated, there needs to be enough memory for the 4 parts of the unallocated header: size, leftsize, right, and left. That being said, this makes the minimum size of data storage in an allocated chunk 16 bytes.

pointer is pointed to the datafield,

Malloc:

allocate:

no need to break chunk into two

need to break

memory cant fulfill, allocate new chunk{

two chunk adjacent{last\_block is unallocated, last\_block is allocated}

not adjacent

}

deallocate:

coalesce with both sides

with left

with right

no colescing

Shell:

fork = 0 in the child process

fork = pid > 0 in the parent process

fork < 0, error

waitpid (pid, NULL), wait for child process who has pid to terminate

dup2(fd1, fd2)

After dup2(fd1, fd2), redirect fd2 to fd1

pipe(int fdpipe[2]) what is write into fdpipe[1] can be read from fdpipe[0]

Regex

+ 1 or more occurrence

. anything

\* 0 or more occurrence

([0-9])\1+ 1 or more matches

([0-9])([0-9])(\1\2)+ 121212 232323

${@:$#} $@ arglist, :$# at index #

${#foo} = 5

[a-z]{3,}

$? is used to find the return value of the last executed command. Try the following in the shell:

Touch update timestamp of a file

Regex

^ match the beginning of a string

$ match the end of a string

? match 0 or 1 times

[0-9]{3} match 3 times

([0-9])\3+ match 3 or more times

If you want parent to wait for child process waitpid (ret) in if (fork)