CSCE 313
LAB session

Wei Zhang

Lab policies

- No need to sign-in.
- You can choose any environment for code compiling.
- Machine problems (MPs)
  - Individual work;
  - Vocareum built-in capabilities will be used on all of them to detect plagiarism.
  - Late submission
    - Refer to the syllabus

Machine Problems (MP's)

- Total 9 MP's of varying complexity
- Some MP's may also allocate opportunities for bonus points
- MP's are designed to stay in sync with subject matter covered in class
- MP 1 and 2 are designed to get the students ramped up on understanding C memory allocation, pointer arithmetic, etc.
  - Rest of the MP's leverage class learning

Machine Problems

ID	Machine Problem	Key Learnings	Complexity
MP 1	High Performance Linked List	C++ refresh, cost of system calls	LOW
MP 2	Memory Allocator	Memory Management	MED-HIGH
MP 3	System Calls and Critical OS Functions	Inner workings of some key system commands	LOW
MP 4	UNIX Process	Anatomy and Attributes of a UNIX process	LOW
MP 5	UNIX Shell	Creation and Execution of a Unix Shell, basic functions	MED

Machine Problems (contd.)

ID	Machine Problem	Key Learnings	Complexity
MP 6	Scheduler	Scheduling Policies	MED
MP 7	Threaded Client-Server	Threading	LOW
MP 8	Advanced Client-Server	Threading, Synchronization	MED
MP 9	IPC Mechanisms	Threading, Synchronization, IPC Mechanisms	MED

MP Teams

- Team Membership
  - You are expected to work in teams of 2
    - Team memberships must be finalized by the **end of this week**
  - Any unresolved situations will need to be brought to the Instructor's notice for quick resolution given the brevity of MP schedule

MP Code Turnin

- Students must strictly follow the instructions provided in Machine Problem statement
  - It is difficult to grade unless instructions are followed
- Code must be turned in on **Vocareum** available on vocareum.com by 11:59pm the due date

MP Grading

- Grading Rubric will be published for each MP on release
- Prelim grading for functionality will be done automagically in Vocareum. Final Grading (enhanced testing, code demo, code quality, report) will be done in the lab. Extent will be determined on a case-by-case basis.
- Both team members must be present in the lab for grading and must be prepared to explain their work as requested by the TA
- TA's reserve the right to grade some aspects of machine problems (e.g. reports) offline

MP Help Resources

1. gdb debugging tool
   - Look at https://www.cs.cmu.edu/~gilpin/tutorial/
2. Lab meetings
   - Most effective (face-to-face)
3. Office hours of PTs
   - All over the week and also some in weekend
4. Piazza Discussion
   - Frequently Asked Questions (1)
   - New Questions/Discussion (2)
5. ~~Email~~
   - ~~Hard for the teaching staff, when helping with code~~

MP Submission

- **Vocareum**
  - New platform
  - Sign-up and be familiar (vocareum.com)
  - Open to your input
- Grades through eCampus
  - Traditional (subject to change depending on our comfort with Vocareum)

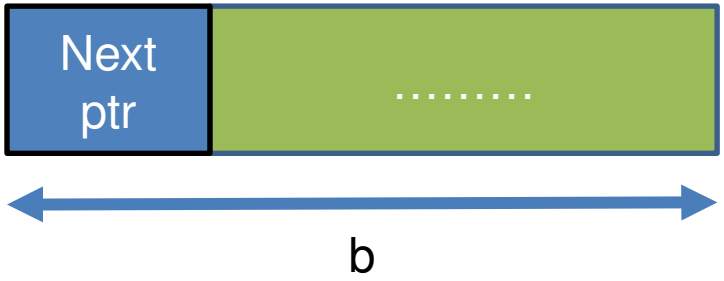
MP1

- Single-Linked List
  - new/malloc a memory block with size *b*.
  - Insert it into the list.
  - Remove one from the list.
  - delete/free its memory.
- New/delete is slow/expensive
  - We reduce # of them to our best.

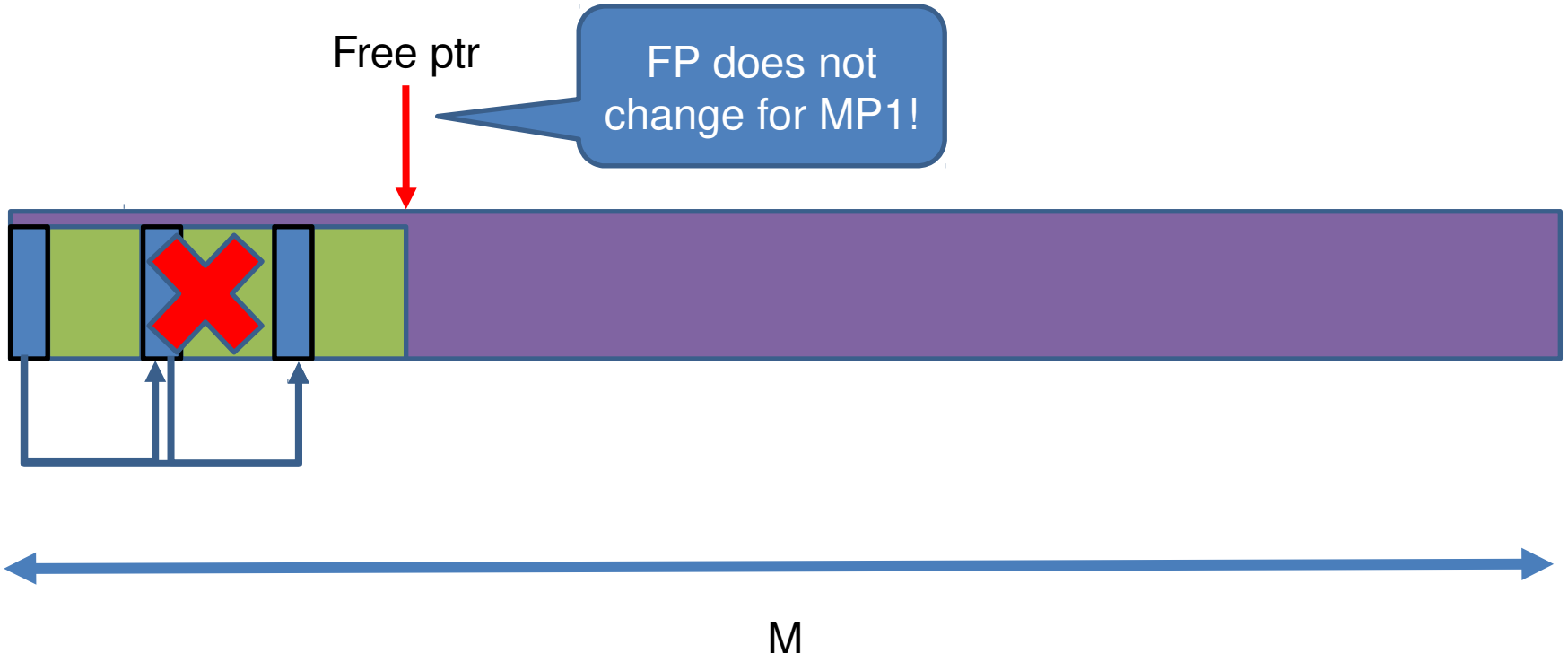
Idea

- Allocate a big block of memory with size *M*.
  - *M* must be integral multiples of the block size *b*.
- Then when inserting a block to the list.
  - Get a block of size *b* from the memory *M*.
  - Connect the block to the list.
- When removing a block
  - Disconnect the block.
  - That's it!
- Deallocate the entire memory when exiting.
- Only one pair of new/delete is needed!

Example - insert



Example - delete



Implementation

- Three files in your program: main.c/cpp, linkedlist.h, and linkedlist.c/cpp.
  - main.c is provided.

- Six functions
  - void Init (int M, int C)
  - void Destroy ( )
  - int Insert (int x , char * valueptr , int valuelen)
  - void Delete (int x)
  - char * Lookup (int x)
  - void PrintList ( )

Implementation

- Generate a program called “testlist”
  - ./testlist [-b <blocksize>] [-s <memsize>]
    - Two optional parameters, if not given, blocksize=128 Bytes, memsize=512KBytes.
    - Use getopt() to parse them.
      - http://www.gnu.org/software/libc/manual/html_node/Example-of-Getopt.html

Questions?

Background

- C language
  - Basic variables: **char** (8 bits), **short** (16), **int** (32), **long** (64), (**unsigned** keyword)
    - sizeof(int)
    - int a; long b; unsigned char c;
  - Control: **if** … **else** (**if**)…, **switch** … **case** … **default**, **for** and **while** loop
    - dead loop: while(1) or for(;;);
  - Array[]
    - char a[10], where a is the address of the array.

Background

- C language
  - Pointer* (store the address)
    - int* p
    - & to get the address, e.g. int a=1; int* p = &a
    - * to access the data in that address, e.g. *p = 2; so now a=2.
    - Pointer can point to a function, e.g. void (*p)(int, char*) defines a pointer to a function like void func(int a, char* b); //useful in Linux kernel development
    - int (*p)[4] vs int *p[4]?
    - Now we have an array like “int a[6]”, how to define a pointer to it, such than we can use the pointer to read/write it?
    - Now we have an array like “int b[6][4]”, how to define a pointer to it, such than we can use the pointer to read/write it?
    - int *p = a;
    - int *p = b, or int (*p)[4] = b; (*(p+1))[2] = 3 or p[1][2] = 3;
      // p+1 means that the address + 4*sizeof(int)
    - Double pointer int **p, a pointer to a pointer. When to use?

Background

- C language
  - Struct
    - struct student{ int netid; char name[64] ; …};
  - Function()
    - int func(int a, char* b) { return 0; } //note, function name is also the address of the function, so we can let p = func;
    - main(); //entry point of the program
    - printf("some string %d\n", aaa); //aaa is an integer variable
  - Definition
    - #define MAX 100 //note, there is no semicolon here
  - Typedef
    - typedef unsigned char byte;
    - typedef struct student student_t;
    - typedef void (*FUNC)(int, char*); //so FUNC f defines a function pointer.
  - Comment
    - /* */ cannot be nested
    - //
    - #if 0 … #endif

Background

- Compiler (compile and link)
  - gcc/g++: C compiler/C++ compiler
  - gcc/g++ -o exename file.c , only one source file
  - more than one source files
    - gcc/g++ -c -g file1.c (compile) //-g is adding debug information
    - gcc/g++ -c -g file2.c (compile)
    - gcc/g++ -o exename file1.o file2.o (link)
- Makefile
  - Automatically call the instructions above in a smarter manner (incremental compile).
    - make -f makefile_name
  - automake, cmake, scons etc are tools for auto-generating makefile.
- Debugger
  - gdb
- Object file inspector
  - objdump //not useful for us

Background

- Bash (Command line environment on Unix/Linux)
  - Use “command --help” to get the command options
  - ls (-l) (list)
  - cd (change directory)
  - pwd (print working directory)
  - ps -ef (to get the #pid of a process, i.e. a running program)
  - kill #pid
  - vi, emacs (terminal based editor)
  - make
- IDE
  - Eclipse CDT
  - Notepad++