**CS241 – 11 Memory Allocators & Discussion**

Problem: After a while, many *malloc*s and *frees* can cause fragmentation of free space (inefficient use of memory resource; harder to find appropriate space for next malloc).

Different algorithms attempt to solve this using different heuristics.

Case study: SLAB allocator for kernel objects

Case study: Buddy allocator

*Terminology*: External Fragmentation: When the available space is not contiguous. Depends on pattern of allocations and frees. vs

Internal Fragmentation: ‘Hidden unused space’ inside each allocation   
(standard example: round up each allocation request to 2^n => unused space *inside* each block)

pThreads

Today:

pthread\_create

pthread\_join

pthread\_exit

1 My program calls pthread\_create twice. How many stacks does my process have?

2 What is the difference between a process and a thread?

3 What does pthread\_cancel do?

and are there alternatives?

4 What is the difference between exit() and pthread\_exit()?

5 Why would you call pthread\_exit in your main method?

6 Give four ways that a thread can be terminated

7 What is the purpose of pthread\_join?

8 What happens if you don't call pthread\_join?

9 start is a temporary variable, so is the following code valid?

int start\_threads() {

int start = 42;

pthread\_create(&tid, 0, myfunc, &start);

}

How could it be made valid?

10 What's wrong with the following code? How can we fix it?

void\* myfunc(void\*ptr) {

}

int main() {

// Each thread gets a different value of i to process

pthread\_t tid;

for(int i =0; i < 10; i++) {

pthread\_create(&tid, 0, myfunc, &i);

}

...

11 Why are some functions e.g. asctime,getenv, strtok, strerror not thread-safe?

char\* to\_message(int num) {

char static result [256];

if(num < 1000) sprintf(result, "%d : blah blah" , num);

else strcpy(result, "Unknown");

return result;

}

12. What are condition variables, semaphores, mutexes?

13. Advantages of threads over forking processes?

14. Can you fork a process with multiple threads?

15. Examples of why you might fork processes