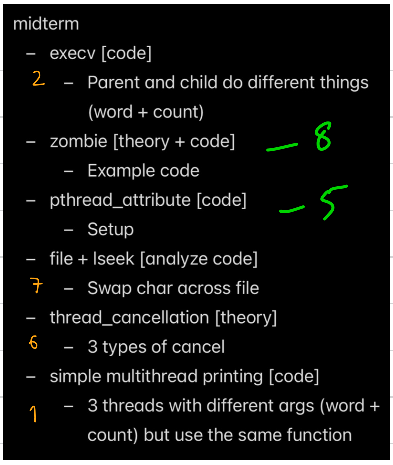


midterm

- execv [code]
  - 2 Parent and child do different things (word + count)
- zombie [theory + code] — 8
  - Example code
- pthread_attribute [code] — 5
  - Setup
- file + lseek [analyze code]
  - 7 Swap char across file
- thread_cancellation [theory]
  - 6 3 types of cancel
- simple multithread printing [code]
  - 1 3 threads with different args (word + count) but use the same function

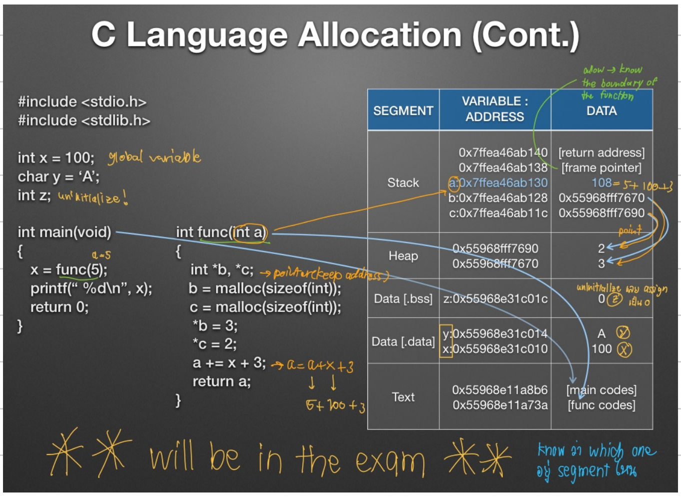
Pass data Argument ?

1. Code multithread 3 threads สร้างคำกะจน.ครั้งต่างกัน รับ argument เข้ามา ?

2. Code multiprocess ทำพ่อ, ลูก พ่อให้ปริ้นอันนึง ส่วนลูกให้ไป exec อีก program นึงที่ปริ้นอีกอย่างนึง

3. Data Segment ให้ code จะถามว่าแต่ละตัวแปรอยู่ segment ไหน <stack, heap, ...>

Text Segment stores instructions of the process
• Data Segment stores
• initialized static data : [.data]
// char name[] = "bob";
• uninitialized static data : [.bss] (Block Started by Symbol)
// int array[100];
• Heap is for dynamic memory demand // malloc();
• Stack is for function call storage and automatic variables



C Language Allocation (Cont.)

```
#include <stdio.h>
#include <stdlib.h>

int x = 100;
char y = 'A';
int z;

int main(void)            int func(int a)
{                         {
   x = func(5);              int *b, *c;
   printf(" %d\n", x);       b = malloc(sizeof(int));
   return 0;                 c = malloc(sizeof(int));
}                            *b = 3;
                             *c = 2;
                             a += x + 3;
                             return a;
                          }
```

SEGMENT	VARIABLE : ADDRESS	DATA
Stack	0x7ffea46ab140 0x7ffea46ab138 a:0x7ffea46ab130 b:0x7ffea46ab128 c:0x7ffea46ab11c	[return address] [frame pointer] 108 0x55968fff7670 0x55968fff7690
Heap	0x55968fff7690 0x55968fff7670	2 3
Data [.bss]	z:0x55968e31c01c	0
Data [.data]	y:0x55968e31c014 x:0x55968e31c010	A 100
Text	0x55968e11a8b6 0x55968e11a73a	[main codes] [func codes]

global variable
uninitialize!
a=5
→ pointer (keep address)
→ a = a+x+3
5+100+3
allow → know the boundary of the function
= 5+100+3
point
uninitialize และ assign เป็น 0
z
y
x
** will be in the exam **
know ว่า which one อยู่ segment ไหน

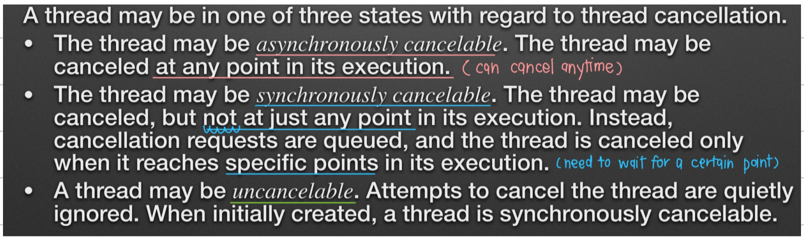
4. Process & Thread ต่างกันยังไง

For some programs that benefit from concurrency, the decision whether to use processes or threads can be difficult. Here are some guidelines to help you decide which concurrency model best suits your program:
• All threads in a program must run the same executable.A child process, on the other hand, may run a different executable by calling an exec function.
• An errant thread can harm other threads in the same process because threads share the same virtual memory space and other resources. For instance, a wild memory write through an uninitialized pointer in one thread can corrupt memory visible to another thread. An errant process, on the other hand, cannot do so because each process has a copy of the program's memory space.
• Copying memory for a new process adds an additional performance overhead relative to creating a new thread. However, the copy is performed only when the memory is changed, so the penalty is minimal if the child process only reads memory.
• Threads should be used for programs that need fine-grained parallelism. For example, if a problem can be broken into multiple, nearly identical tasks, threads may be a good choice. Processes should be used for programs that need coarser parallelism.
• Sharing data among threads is trivial because threads share the same memory. (However, great care must be taken to avoid race conditions.) Sharing data among processes requires the use of IPC mechanisms. This can be more cumbersome but makes multiple processes less likely to suffer from concurrency bugs.

5. Thread attribute การสร้าง 5 ขั้นตอน มีไรบ้าง + code

To specify customized thread attributes, you must follow these steps:
1. Create a pthread_attr_t object. The easiest way is simply to declare an automatic variable of this type.
2. Call pthread_attr_init, passing a pointer to this object. This initializes the attributes to their default values.
3. Modify the attribute object to contain the desired attribute values.
4. Pass a pointer to the attribute object when calling pthread_create.
5. Call pthread_attr_destroy to release the attribute object. The pthread_attr_t variable itself is not deallocated; it may be reinitialized with pthread_attr_init.

6. อธิบาย cancel type ทั้ง 3 อัน มีไรบ้าง เป็นยังไง



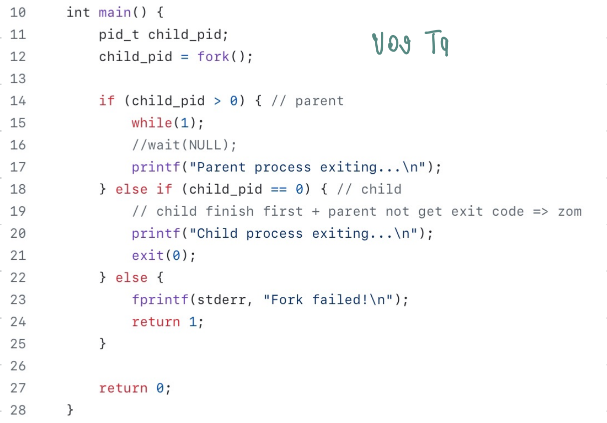
A thread may be in one of three states with regard to thread cancellation.

- The thread may be *asynchronously cancelable*. The thread may be canceled at any point in its execution. (can cancel anytime)
- The thread may be *synchronously cancelable*. The thread may be canceled, but not at just any point in its execution. Instead, cancellation requests are queued, and the thread is canceled only when it reaches specific points in its execution. (need to wait for a certain point)
- A thread may be *uncancelable*. Attempts to cancel the thread are quietly ignored. When initially created, a thread is synchronously cancelable.

7. Lseek ให้ code มา มี file 2 อัน เขียนโค้ดให้คำ 2 คำ read/write สลับกัน (คำแปลกๆ 2 ไฟล์ มันจะมารวมเป็น you can do everything กะ but not everything!)

- output ไฟล์ 1 กะไฟล์ 2 คืออะไร / อธิบายภาพรวม program ทำอะไร / line: int n = seek set ไปต่อท้าย ของไฟล์ ทำเพื่ออะไร

8. zombie theory + code



ของ Ta

```
int main() {
    pid_t child_pid;
    child_pid = fork();

    if (child_pid > 0) { // parent
        while(1);
        //wait(NULL);
        printf("Parent process exiting...\n");
    } else if (child_pid == 0) { // child
        // child finish first + parent not get exit code => zom
        printf("Child process exiting...\n");
        exit(0);
    } else {
        fprintf(stderr, "Fork failed!\n");
        return 1;
    }

    return 0;
}
```

- หา Answer 2
- จำ Answer 3
- ทำ code practice 4
- อ่านslide (ละเอียด) 1

ส & อ

พฤ & ศ