HỆ ĐIỀU HÀNH BÀI 9

Câu 1: API nào sau đây được sử dụng để đợi trên Mutex khi tiến trình muốn truy cập vào vùng tài nguyên dùng chung?

A. WaitForSingleObject()
B. CloseHandle()
C. CreateMutex()
D. ReleaseMutex()

Câu 2: API nào sau đây được sử dụng để tiến trình giải phóng các tài nguyên được bảo vệ Mutex?

A. ReleaseMutex()
B. CloseHandle()
C. WaitForSingleObject()
D. SetPriorityClass()

Câu 3: API nào được sử dụng để tạo tiến trình trong Windows?

A. Create Thread()
B. InitProcess()
C. StartProcess()
D. CreateProcess()

Câu 4: Hàm nào được sử dụng để hủy tiến trình trong Windows?

A. TerminateProcess()
B. Kill Process()
C. ExitProcess()
D. CloseProcess()

Câu 5: Việc rò rỉ bộ nhớ thường xảy ra ở vùng nhớ nào nhất khi cấp cho tiến trình?

A. Data
B. Heap
C. Stack
D. Code

Câu 6: Khi sử dụng API SetPriorityClass(hProcess, HIGH_PRIORITY_CLASS) cho một tiến trình, điều gì xảy ra?

A. Tiến trình bị giảm quyền truy cập CPU
B. Tiến trình chuyển sang trạng thái dừng
C. Tiến trình bị hủy ngay lập tức
D. Tiến trình được ưu tiên sử dụng CPU nhiều hơn

Câu 7:Công cụ nào sau đây có thể sử dụng để quan sát các vùng nhớ cho từng tiến trình?

A. Windows Memory Diagnostic
B. Resource Monitor
C. Task Manager
D. VMMap

Câu 8: Hậu quả của việc hủy tiến trình bất ngờ là gì?

A. Hệ điều hành tự động phục hồi tiến trình
B. Tiến trình được lưu lại để chạy lần sau
C. CPU bị giảm hiệu suất ngay lập tức
D. Mất dữ liệu, rò rỉ bộ nhớ

Câu 9: Lệnh nào sau đây có thể gây dò rỉ bộ nhớ?

A. int* heap = (int*)malloc(10* sizeof(int));
B. float y=2.5;
C. char ch='A
D. int x =10;

Câu 10:Tại sao việc thay đổi độ ưu tiên của tiến trình là quan trọng?

A. Giúp tiến trình chạy nhanh hơn mọi lúc
B. Giảm xung đột giữa tiến trình và hệ điều hành
C. Giúp CPU hiệu quả hơn, tăng hiệu suất xử lý các chương trình quan trọng
D. Giúp tiến trình sử dụng ít RAM hơn

Câu 11:Điều gì xảy ra khi một chương trình ghi vào vùng nhớ chưa được cấp phát?

A. Không có vấn đề gì xảy ra
B. Hệ điều hành tự động cấp phát vùng nhớ

C. Lỗi lỗi truy cập bộ nhớ không hợp lệ
D. Chương trình chạy nhanh hơn

Câu 12: Hằng số nào thể hiện mức độ ưu tiên cao nhất của một tiến trình?

A. NORMAL_PRIORITY_CLASS
B. ABOVE_NORMAL_PRIORITY_CLASS
C. HIGH_PRIORITY_CLASS
D. IDLE_PRIORITY_CLASS

Câu 13: Hiện tượng tràn bộ nhớ xảy ra trong trường hợp nào sau đây:

A. Tiến trình truy cập ra ngoài phạm vi của máng
B. Một vòng lập chạy vô hạn không giải phóng bộ nhớ
C. Tiến trình truy xuất đến địa chỉ bộ nhớ chưa được cấp phát
D. Tiến trình ghi dữ liệu vào vùng nhớ có kích thước nhỏ hơn số byte cần ghi

Câu 14: Đâu là các vấn về bộ nhớ mà người lập trình cần chú ý viết một chương trình (chọn 2)?

A. Vấn đề rò rỉ bộ nhớ
B. Vấn đề về khởi tạo giá trị cho các biến cục bộ
C. Vấn đề tràn bộ nhớ
D. Vấn đề về khởi tạo giá trị cho các biến toàn cục

Câu 15: Cho đoạn code sau:

```c
#include <windows.h>
#include <stdio.h>
int main() {
    STARTUPINFO si;
    PROCESS_INFORMATION pi;
    ZeroMemory(&si, sizeof(si));
    si.cb = sizeof(si);
    ZeroMemory(&pi, sizeof(pi));
    if (CreateProcess(TEXT("C:\\Windows\\System32\\notepad.exe"),
                      NULL,
                      NULL,
                      NULL,
                      FALSE,
                      0,
                      NULL,
                      NULL,
                      &si,
                      &pi))  {
        CloseHandle(pi.hProcess);
        CloseHandle(pi.hThread);
        }
    return 0;
}
```

Hãy cho biết đoạn code dùng để làm gì?

A. Hủy một tiến trình đang chạy (notepad.exe) trong Windows
B. Đồng bộ giữa hai tiến trình notepad.exe trong Windows
C. Tạo một tiến trình notepad.exe trong Windows
D. Thay đổi độ ưu tiên của tiến trình đang chạy (notepad.exe) trong Windows

Câu 16: Chương trình sau có gây rò rỉ bộ nhớ không?

```cpp
#include <iostream>

void memoryLeak() {
    int* ptr = new int[100];
}

int main() {
    while (true) {
        memoryLeak();
    }
    return 0;
}
```

A. Có, nhưng chỉ khi chương trình chạy lâu dài
B. Có, vì con trỏ chưa được giải phóng khi kết thúc chương trình
C. Không vì con trỏ đã được giải phóng khi kết thúc chương trình
D. Không, vì không có sự cấp phát bộ nhớ nào

Câu 17: Cho đoạn chương trình làm việc với tiến trình trong Windows:

```cpp
HANDLE hMutex = CreateMutex(NULL, FALSE, TEXT("MyMutex"));
DWORD waitResult = WaitForSingleObject(hMutex, INFINITE);

if (waitResult == WAIT_OBJECT_0) {
    Sleep(5000);
    ReleaseMutex(hMutex);
} else {
    CloseHandle(hMutex);
}
CloseHandle(hMutex);
```

Hãy cho biết chương trình trên thực hiện khi nào?

A. Tạo một tiến trình cho một ứng dụng bất kỳ trong Windows
B. Hủy một tiến trình đang chạy trong Windows
C. Đồng bộ giữa các tiến trình chạy trong Windows

D. Thay đổi độ ưu tiên của tiến trình đang chạy trong Windows

Câu 18: Chương trình sau có xảy ra vấn đề về bộ nhớ không?

```
#include <iostream>

void memoryLeak() {
    int* ptr = new int[100];
    free(ptr);
    ptr = NULL;
}

int main() {
    while (true) {
        memoryLeak();
    }
    return 0;
}
```

A. Có, xảy ra rò rỉ bộ nhớ
B. Có, xảy ra vấn đề tranh chấp tài nguyên
C. Không, chương trình chạy bình thường
D. Có, xảy ra lỗi tràn bộ nhớ

Câu 19: Chương trình sau có thể gây ra lỗi gì?

```
#include <stdio.h>

int main() {
    int a[5];
    a[10]=100;
    return 0;
}
```

A. Truy cập bộ nhớ không hợp lệ (Segmentation fault)
B. Không có lỗi gì
C. Rò rỉ bộ nhớ (Memory Leak)
D. Tràn bộ nhớ (Buffer Overflow)

Câu 20: Chương trình sau gặp lỗi gì?

```
1
2   #include<stdlib.h>
3   int main() {
4       int* ptr = NULL;
5       *ptr = 200;
6       return 0;
7   }
```

A. Truy cập bộ nhớ không hợp lệ (Segmentation fault)
B. Tràn bộ nhớ (Buffer Overflow)
C. Rò rỉ bộ nhớ
D. Không có lỗi

Câu 21: Chương trình sau có thể dẫn đến lỗi gì?

```
1   #include <stdio.h>
2
3   void recursive() {
4       int arr[100000];
5       recursive();
6   }
7
8   int main() {
9       recursive();
10      return 0;
11  }
```

A. Không có vấn đề về bộ nhớ nhưng chạy vô hạn
B. Lỗi rò rỉ bộ nhớ
C. Không có lỗi gì
D. Lỗi tràn bộ nhớ (Stack Overflow)

Câu 22:

Chương trình sau đây có lỗi gì?

```
void func(int n) {
    int arr[200000];
    func(n + 1);
}

int main() {
    func(1);
    return 0;
}
```

A. Rò rỉ bộ nhớ (Memory leak)
B. Truy cập bộ nhớ không hợp lệ (Segmentation fault)
C. Tràn bộ nhớ (Stack Overflow)
D. Không có lỗi gì

Câu 23: Cho đoạn chương trình làm việc với tiến trình trên Windows như sau:

```
HANDLE hProcess = GetCurrentProcess();
if (SetPriorityClass(hProcess, HIGH_PRIORITY_CLASS)) {
    printf(" Thanh cong! \n");
} else {
    printf(" Loi: %lu\n", GetLastError());
}
```

Hãy cho biết đoạn chương trình trên thực hiện thao tác gì?

A. Tạo một tiến trình cho một ứng dụng bất kỳ trong Windows
B. Thay đổi độ ưu tiên của tiến trình đang chạy trong Windows
C. Đồng bộ giữa hai tiến trình chạy trong Windows
D. Hủy một tiến trình đang chạy trong Windows

Câu 24: Cho đoạn chương trình làm việc với tiến trình trong Windows:

```
void KillProcess(HANDLE hProcess) {
    if (TerminateProcess(hProcess, 0)) {
        printf("Thanh cong! \n");
    } else {
        printf("Loi: %lu\n", GetLastError());
    }
    CloseHandle(hProcess);
}
```

Hãy cho biết đoạn chương trình trên dùng để làm gì?

A. Hủy một tiến trình đang chạy trong Windows dựa vào Handle
B. Thay đổi độ ưu tiên của tiến trình đang chạy dựa vào Handle
C. Đồng bộ giữa hai tiến trình chạy trong Windows dựa vào Handle
D. Tạo một tiến trình cho một ứng dụng bất kỳ trong Windows

Câu 25: Kéo các API hoặc toán tử khi làm việc với tiến trình trong Windows dưới đây vào ô tương ứng với mô tả của chúng:

sizeof() GetCurrentProcess()

CloseHandle() ZeroMemory()

Xóa sạch vùng nhớ của một cấu trúc dữ liệu	
Cấp phát vùng nhớ cho biến hoặc cấu trúc dữ liệu	
Đóng và giải phóng tài nguyên được sở hữu bởi một tiến trình	
Lấy về Handle của tiến trình hiện hành	

Câu 26:Kéo các API làm việc với tiến trình trong Windows dưới đây vào ô tương ứng với mô tả của chúng:

TerminateProcess() SetPriorityClass()

CloseHandle() CreateProcess()

Đóng Handle của tiến trình đã tạo và giải phóng tài nguyên bị chiếm bởi tiến trình	
Kết thúc một tiến trình	
Tạo một tiến trình	
Thay đổi độ ưu tiên của tiến trình	

Câu 27:

Hãy kéo các câu lệnh gây ra các vấn đề về bộ nhớ tương ứng với các mô tả dưới đây:

char buffer[3]; strcpy(buffer, "Cong nghe thong tin");

int a[4]; a[5] = 100;

lint* heap = (int*)malloc(10*sizeof(int)); heap = NULL;

Truy cập bộ nhớ không hợp lệ	
Rò rỉ bộ nhớ	
Tràn bộ nhớ	

Câu 28:Kéo các API khi làm việc với tiến trình trong Windows dưới đây vào ô tương ứng với mô tả của chúng:

GetLastError() ReleaseMutex()

CreateMutex() WaitForSingleObject()

Tạo một Mutex để đồng bộ tiến trình	
Trả về mã lỗi nếu một thao tác thực hiện không thành công	
Tiến trình đợi trên Mutex để truy cập vào vùng tài nguyên được chia sẻ	
Tiến trình giải phóng tài nguyên được bảo vệ bởi Mutex	

Câu 29: Cho khai báo sau đây trong C:

```
char buffer[10];
```

Lệnh nào sau đây gây ra hoặc không gây ra lỗi tràn bộ nhớ? Kéo các đáp án vào ô tương ứng:

strcpy(buffer, "Lap Trinh!")

strcpy(buffer, "Chao ban!")

strcpy(buffer, "Cong nghe thong tin - ictu - thai nguyen")

strcpy(buffer, "Nguyen ly he dieu hanh - cong nghe thong tin!")

1) Các lệnh KHÔNG gây tràn bộ nhớ:	1) Các lệnh gây ra tràn bộ nhớ:

Câu 30: Cho đoạn chương trình C như dưới đây. Hãy viết lệnh sử dụng hàm free vào vị trí--------- để giải phóng bộ nhớ được cấp phát cho biến arr để tránh vấn đề rò rỉ bộ nhớ:

```
1  void leak() {
2      int* arr = new int[100];
3      ----------  ;
4  }
```

1) Lệnh cần viết: