CSE312 Hw1 Report

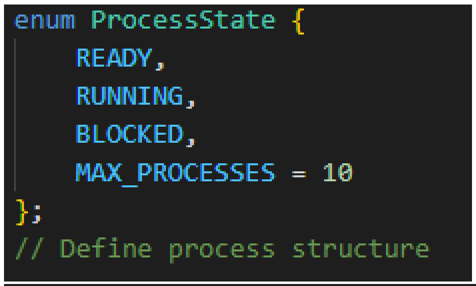
Elifnur Kabalcı

1801042617

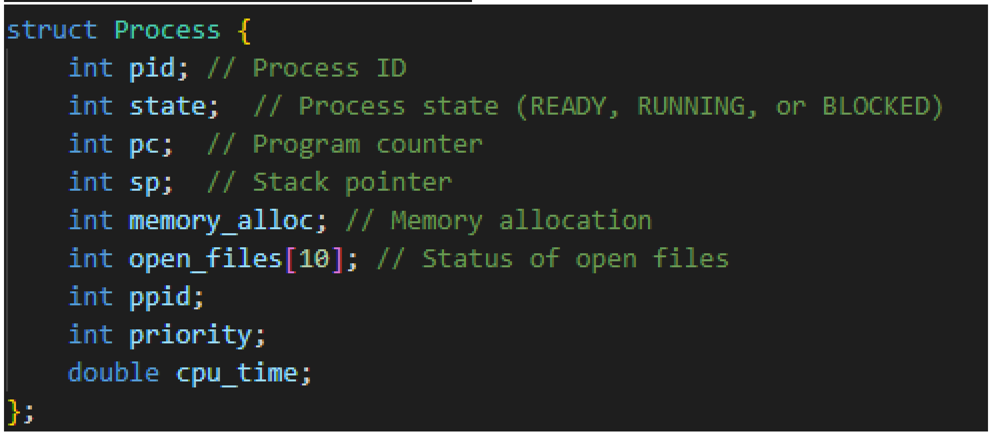
I made additions on top of the 15th video in the video series. I will go through the areas where I made additions one by one. I installed and ran a nested virtual machine. but I couldn't run the own code.

a) Process.h process.cpp
   I created the process.h and process.cpp files in order to store the data of the Process table and create processes. In the process.h file, I kept the process state information. I defined the Process struct and made function definitions. In the functions in the process.cpp file, I created processes, updated the state of the process, and wrote a function that can retrieve the process id of the process.

   1)process.h screnshoots



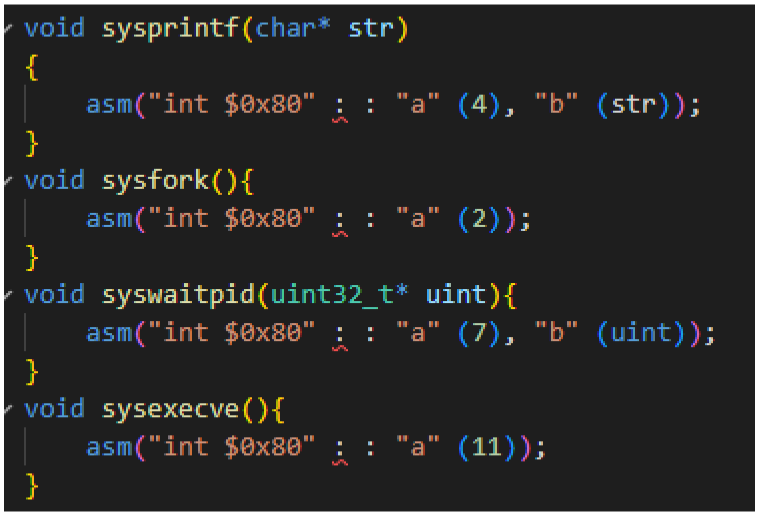
```
enum ProcessState {
    READY,
    RUNNING,
    BLOCKED,
    MAX_PROCESSES = 10
};
// Define process structure
```



```
struct Process {
    int pid; // Process ID
    int state;  // Process state (READY, RUNNING, or BLOCKED)
    int pc;  // Program counter
    int sp;  // Stack pointer
    int memory_alloc; // Memory allocation
    int open_files[10]; // Status of open files
    int ppid;
    int priority;
    double cpu_time;
};
```

b) kernel.cpp
   In the video series, a numerical value was given for each process operation, and it was also indicated whether it takes eax, ebx, and ecx values in this table. In the kernel file, sysprintf was used to send the eax and ebx values. Among these values, the value of eax was sent as 4. Therefore, I assigned 2 for fork, 7 for waitpid, and 11 for execve. The waitpid command takes an unsigned integer as its b value. I defined the functions that can perform operations, like printf, in the early parts of the file, but I couldn't fill them.

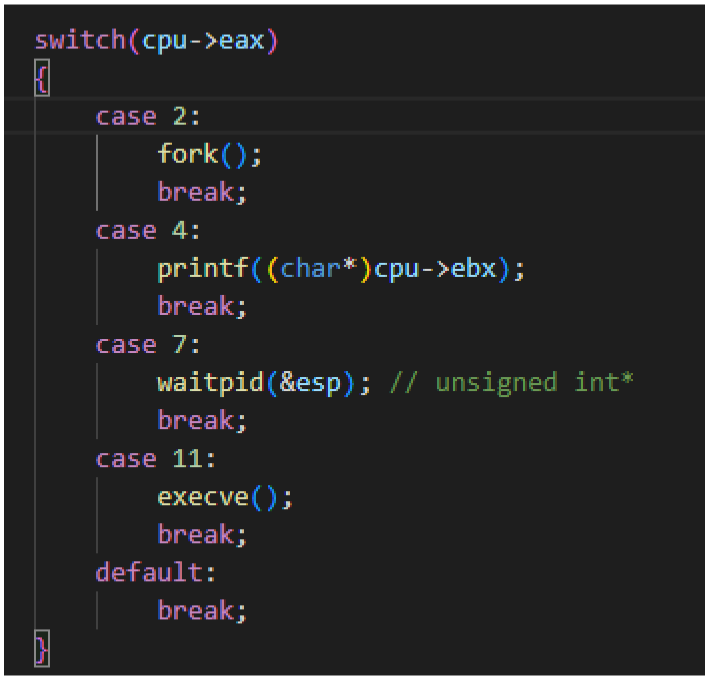
2)Screenshot form kernel.cpp



```
void sysprintf(char* str)
{
    asm("int $0x80" : : "a" (4), "b" (str));
}
void sysfork(){
    asm("int $0x80" : : "a" (2));
}
void syswaitpid(uint32_t* uint){
    asm("int $0x80" : : "a" (7), "b" (uint));
}
void sysexecve(){
    asm("int $0x80" : : "a" (11));
}
```

c) syscall.cpp

In the syscall section, we are keeping the eax and ebx values sent from kernel.cpp with handle interrupt. That's why I included them in the switch case. In this section, I thought that an assembly file written for the asm file should be called, but I couldn't find a specific asm file defined in the file. The interrupt handler page was a general expression. I think the connection error is in this section, but I couldn't solve the error.



```
switch(cpu->eax)
{
    case 2:
        fork();
        break;
    case 4:
        printf((char*)cpu->ebx);
        break;
    case 7:
        waitpid(&esp); // unsigned int*
        break;
    case 11:
        execve();
        break;
    default:
        break;
}
```

d) last

In the video series, it is mentioned that the schedule function written is round robin scheduling. I wanted to apply the mentioned algorithms. However, I did not add them because I thought I had to link these applications to possix methods. I added necessary objects for new files.