2020105695_김희성

Scheduling Queues

PCB Implementation

Code: PCB.h, PCB.cpp

```
enum class Status
{
    new_,
    ready,
    running,
    waitting,
    terminate
};

enum class Priority
{
    low,
    middle,
    high
};

enum class IOStatus
{
    no,
    exist,
    doing,
    done
};

class ProcessControlBlock
{
    static int PCB_num;
    int process_id;
    IOStatus have_IO;
    bool is_in_RAM;
    Priority priority;
    Status state;
    ProcessControlBlock* next_pcb;
    char const environment_variable[256] = { 0 };

public:
    //explicit ProcessControlBlock();
    explicit ProcessControlBlock(const Priority prior, const IOStatus has_IO);
    explicit ProcessControlBlock(const ProcessControlBlock& PCB);
    ~ProcessControlBlock();
    void PrintStatus() const;
    void SetStatus(const Status stat);
    IOStatus HaveIO()const;
    void SetHaveIO(const IOStatus b);
    bool IsInRam()const;
    void SetRamState(const bool b);
    int GetPID()const;
    Priority GetPrior()const;
    void SetPrior(const Priority& p);
    void SetNextPointer(ProcessControlBlock* const ptr_Pcb);
    ProcessControlBlock* GetNextPointer() const;
};
```

차별점:

- window에선 환경 변수가 부모 프로세스에서 자식 프로세스로 상속된다고 나와있었다. 그래서 환경 변수에 관한 term도 넣었다. 총 프로세스의 개수와 process id 등등 많은 정보를 담았다.
- Priority 정보와 IO 정보가 scheduling 도중 바뀐다. Ex) ready queue에서는 ready 상태, IO 상태에선 waiting 상태로 바뀐다.

보완점:

- Context switching 과정에서 register(SP, PC) 등의 실행 중인 process가 기록된다고 배웠다. 하지만 이것 까진 구현하지 못했다.
- Job scheduler가 secondary storage에 저장돼야 하지만 RAM에 저장된다.

Scheduling Queues Implementation

Code: scheduler.h, scheduler.cpp

```
class Scheduler
{
    ProcessControlBlock *job_front,    *job_rear;
    ProcessControlBlock *ready_front,  *ready_rear;
    ProcessControlBlock *device_front, *device_rear;

    int job_length; // 전체길이 index는 -1해서 사용.
    int ready_length;
    int device_length;

public:
    explicit Scheduler();
    ~Scheduler();

    // LEVLE1 API
    void LongTermScheduling(); // Job queue-> ready queue
    void ShortTermScheduling();
    void IOScheduling();
    void TimeExeption();
    void LoadPCBs(ProcessControlBlock const& pcbs); //Stack to Heap.. malloc Node
    bool IsEmpty()const;
    //void SortQueue(); // call Job ready device queue sort

    // LEVEL2 API
    //!caution! you must put dynamic allocated PCB
    void JobPush(const ProcessControlBlock& PCB);
    ProcessControlBlock* JobPop(); //If PCB hasn't dispatched, Job QUEUE not deleted.

    void ReadyPush(ProcessControlBlock& PCB); //PCB status chagne, new -> ready, is_in_ram -> true
    ProcessControlBlock* ReadyPop();
    void DevicePush(const ProcessControlBlock& PCB);
    ProcessControlBlock* DevicePop(); // IO true -> false
    void CpuProcess(ProcessControlBlock& PCB); //free PCB, call DeleteJobQueueItem
    void PrintQueue() const; // print Job ready device queue status

    // LEVEL3 API
    void DeleteJobQueueItem(ProcessControlBlock& PCB); //free PCB

};

#endif //scheduler
```

차별점:

- Linked list 기반으로 각각의 queue를 만들었다. 보통의 linked list로 구현된 queue는 push 할 때 동적 할당되고 pop할 때 해제된다. 이동 과정에서 동적 할당을 여러 번 하면 overhead가 클 것 같아서 job Queue에 올릴 때와, Ready Queue에 올릴 때만 동적 할당을 진행했다. (자료구조 같이 수강하고 있어서 확실 한진 모르겠습니다!)

보완점:

- Scheduling하는 과정에서 priority로 round robin하는 과정을 구하지 못했다.
- Priority 기반 scheduling 기능을 추가하고 싶었다. 하지만 sorting 알고리즘을 아직 배우는 중이다.

배운점:

- Container가 node Pointer를 가질 때, constructor에 new연산을 통해 넣어 놓는 것이 좋을 것 같다.
- 복사 생성자는 Stack에서 Heap으로 할당해줄 수 있다. Ram에 load 가능하다.

Report

PBC가 한 개씩 진행되는 것을 Exeption을 통해 나타내봤다.

Code: main.cpp

```cpp
int main()
{
    ProcessControlBlock IO_PCB_(Priority::high, true);
    ProcessControlBlock non_IO_PCB_(Priority::middle, false);
    ProcessControlBlock non_IO_PCB_2(Priority::low, false);

    // scheduler() 왜 안됨? 함수처럼 해석 될 수 있기 때문에 uniform initialize를 사용하여 초기화 한다.
    Scheduler scheduler{};

    // explicit copy 생성자에 ProcessControlBlock a[2] = {obj1, obj2} 안 됨. copy 생성자에 malloc을 사용했기 때문에 암묵적으로 일어나면 위험함.
    //copy constructor를 통해 stack memory의 PBC를 heap으로
    scheduler.LoadPCBs(IO_PCB_);
    scheduler.LoadPCBs(non_IO_PCB_);
    scheduler.LoadPCBs(non_IO_PCB_2);

    while(!scheduler.IsEmpty()) // 다음엔 긍정인 의미의 변수 이름 쓰기.
    {
        scheduler.TimeExeption();
        scheduler.LongTermScheduling();
        scheduler.TimeExeption();
        scheduler.ShortTermScheduling();
        scheduler.TimeExeption();
        scheduler.IOScheduling();
    }
    return 0;
}
```

결과) ->result.txt, scheduling_project.exe

```
queue made
Process constructed. constructed PCB ID : 3
Process constructed. constructed PCB ID : 4
Process constructed. constructed PCB ID : 5
Time Interupt!
press any key and enter
--------------------------------

In Job Queue:
| 3 |  | 4 |  | 5 |

In Ready Queue:


In Device Queue:


--------------------------------
c
Time Interupt!
press any key and enter
--------------------------------

In Job Queue:
| 4 |  | 5 |

In Ready Queue:
| 3 |

In Device Queue:


--------------------------------
c
Start IO PID : 3
Time Interupt!
press any key and enter
--------------------------------

In Job Queue:
| 4 |  | 5 |

In Ready Queue:


In Device Queue:
| 3 |

--------------------------------
c
End IO PID : 3
Time Interupt!
press any key and enter
```