

A thick dark blue vertical bar on the left side of the page, with a blue arrow pointing right from its center, partially overlapping the title.

Memory Management

Using C++

Several thin, curved, light gray lines in the bottom left corner, resembling stylized grass or abstract brushstrokes.

日資工三甲 U0924043 陳皇任

一、編譯環境

(一) 編譯器資訊

Compiler Version:

MinGW: x86_64-8.1.0-posix-seh-rt_v6-rev0

Target: x86_64-w64-mingw32 Thread model: posix

gcc version 8.1.0 (x86_64-posix-seh-rev0, Built by MinGW-W64 project)

(二) 編譯指令

```
$ g++ .\main.cpp -o .\main.exe -g -Wall -std=c++17
```

二、程式碼（因為分檔案撰寫所以將會有多個檔案你不同標題分割）

.\Process.hpp

```

1  #include <list>
2  #include <string>
3  #include "SystemMemory.hpp"
4  #ifndef Process_hpp
5      #define Process_hpp
6      #include "ProcessMemory.hpp"
7      class Process
8      {
9      private:
10     protected:
11     public:
12         std::size_t ID;
13         std::string Name;
14         ProcessMemory ProcessMemory_;
15         SystemMemory* SystemMemory_;
16         Process(std::size_t, std::string, std::size_t, SystemMemory*, std::size_t);
17         void deploy(std::size_t);
18         void print(char*, std::size_t, std::ostream&);
19         std::size_t physicalAddress(std::size_t, bool, bool, char*, std::size_t, std::ostream&);
20         // void print(char*, std::size_t, std::ostream&);
21     };
22 #endif
23 #ifndef Process_cpp
24     #include "Process.cpp"
25 #endif

```

.\Process.cpp

```

1  #include "Debug.hpp"
2  #ifndef Process_hpp
3  #include "Process.hpp"
4  #endif
5  // #include "Processor.h"
6  #ifndef Process_cpp
7  #define Process_cpp
8  Process::Process(std::size_t id, std::string name, std::size_t processMemorySize, SystemMemory* systemMemory, std::size_t initialDeployNumber=0)
9      : ID(id), Name(name), ProcessMemory_(processMemorySize, systemMemory->Frame), SystemMemory_(systemMemory)
10     {
11         // debug_Flag(std::cout);
12         for (std::size_t index = 0; index < initialDeployNumber; index++) {debug_Flag(std::cout);deploy(index);}
13     }
14 void Process::deploy(std::size_t pageId)
15 {
16     // (*std::next(ProcessMemory_.Paging.PagingTable.begin(), pageId))>Valid = true;
17     // (*std::next(ProcessMemory_.Paging.PagingTable.begin(), pageId))>SystemMemoryFrame_ID = SystemMemory->Frame.InvalidList.front()->ID;
18     // SystemMemory->Frame.InvalidList.front()->Valid = true;
19     // SystemMemory->Frame.InvalidList.pop_front();
20     ProcessMemory_.deploy(pageId);
21 }
22 void Process::print(char* Prefix(char*(""), std::size_t shift=0, std::ostream& stream=std::cout)
23 {
24     // debug_Flag(std::cout);
25     for (std::size_t index = 0; index < shift; index++) {stream<<Prefix; stream<<"Process's ID: "<<ID<<std::endl;
26     for (std::size_t index = 0; index < shift; index++) {stream<<Prefix; stream<<"Process's Name: "<<Name<<std::endl;
27     ProcessMemory_.print(Prefix, shift+1);
28 }
29 std::size_t Process::physicalAddress(std::size_t virtualAddress, bool printCalculationProcess=false, bool newline=false, char* Prefix(char*(""), std::size_t shift=0, std::ostream& stream=std::cout)
30 {
31     print((char*)" ", 2);
32     return ProcessMemory_.physicalAddress(virtualAddress, printCalculationProcess, newline, Prefix, shift, stream);
33 }
34 #endif

```

.\ProcessMemory.hpp

```

1  #include "SystemMemory.hpp"
2  #include "SystemMemoryFrame.hpp"
3  #ifndef ProcessMemory_hpp
4  #define ProcessMemory_hpp
5  #include "ProcessMemoryPaging.hpp"
6  class ProcessMemory
7  {
8  private:
9  protected:
10 public:
11     std::size_t Size;
12     ProcessMemoryPaging Paging;
13     SystemMemoryFrame& Frame;
14     // SystemMemory* SystemMemory_;
15     ProcessMemory(ProcessMemory&);
16     ProcessMemory(std::size_t, SystemMemoryFrame&);
17     void deploy(std::size_t);
18     void print(char*, std::size_t, std::ostream&);
19     std::size_t physicalAddress(std::size_t, bool, bool, char*, std::size_t, std::ostream&);
20 };
21 #endif
22 #ifndef ProcessMemory_cpp
23 #include "ProcessMemory.cpp"
24 #endif

```

.\ProcessMemory.cpp

```

1  #include "Debug.hpp"
2  #include <cstring>
3  #ifndef ProcessMemory_hpp
4  #include "ProcessMemory.hpp"
5  #endif
6  #ifndef ProcessMemory_cpp
7  #define ProcessMemory_cpp
8  ProcessMemory::ProcessMemory(ProcessMemory& processMemory)
9  : Size(processMemory.Size), Paging(processMemory.Paging), Frame(processMemory.Frame)
10 {debug_Flag(std::clog);}
11 ProcessMemory::ProcessMemory(std::size_t processMemorySize, SystemMemoryFrame& systemMemoryFrame)
12 : Size(processMemorySize), Paging(processMemorySize, Frame), Frame(systemMemoryFrame)
13 {debug_Flag(std::clog);}
14 void ProcessMemory::deploy(std::size_t pageId)
15 {
16     // (*std::next(Paging.PagingTable.begin(), pageId))->Valid = true;
17     // (*std::next(Paging.PagingTable.begin(), pageId))->SystemMemoryFrame_ID = Frame.InvalidList.front()->ID;
18     // Frame.InvalidList.front()->Valid = true;
19     // Frame.InvalidList.pop_front();
20     // debug_Flag(std::clog);
21     Paging.deploy(pageId);
22 }
23 void ProcessMemory::print(char* Prefix=(char*)"", std::size_t shift=0, std::ostream& stream=std::cout)
24 {
25     for (std::size_t index = 0; index < shift; index++) {stream<<Prefix;} stream<<"Process's Memory Size: "<<Size<<std::endl;
26     Paging.print(Prefix, shift+1);
27 }

```

```

28 std::size_t ProcessMemory::physicalAddress(std::size_t virtualAddress, bool printCalculationProcess=false, bool newline=false, char* Prefix=(char*)"", std::size_t shift=0, std::ostream& stream=std::cout)
29 {
30     std::size_t pageId = virtualAddress/Frame.Size;
31     if (!((*std::next(Paging.PagingTable.begin(), pageId))>Valid)) deploy(pageId);
32     if (printCalculationProcess)
33     {
34         if (newline)
35         {
36             stream<<std::endl;
37             for (std::size_t index = 0; index <= shift; index++) {stream<<Prefix;}
38             stream<<" ";
39         }
40         stream<<"<<<((*std::next(Paging.PagingTable.begin(), pageId))>SystemMemoryFrame_ID)<<"<<Frame.Size<<"<<"
41         <<"<<virtualAddress<<"<<((*std::next(Paging.PagingTable.begin(), pageId))>ID)<<"<<Frame.Size<<"<<"
42         if (newline)
43         {
44             stream<<std::endl;
45             for (std::size_t index = 0; index < shift; index++) {stream<<Prefix;}
46             stream<<" ";
47         }
48         stream<<"<<((*std::next(Paging.PagingTable.begin(), pageId))>SystemMemoryFrame_ID)*Frame.Size<<"<<"
49         <<"<<virtualAddress<<"<<((*std::next(Paging.PagingTable.begin(), pageId))>ID)*Frame.Size<<"<<"
50         if (newline)
51         {
52             stream<<std::endl;
53             for (std::size_t index = 0; index < shift; index++) {stream<<Prefix;}
54             stream<<" ";
55         }
56         stream<<"<<((*std::next(Paging.PagingTable.begin(), pageId))>SystemMemoryFrame_ID)*Frame.Size<<"<<"
57         <<"<<virtualAddress-(((std::next(Paging.PagingTable.begin(), pageId))>ID)*Frame.Size)<<"<<"
58         if (newline)
59         {
60             stream<<std::endl;
61             for (std::size_t index = 0; index < shift; index++) {stream<<Prefix;}
62             stream<<" ";
63         }
64         stream<<(((std::next(Paging.PagingTable.begin(), pageId))>SystemMemoryFrame_ID)*Frame.Size)+(virtualAddress-(((std::next(Paging.PagingTable.begin(), pageId))>ID)*Frame.Size));
65         if (newline)
66         {
67             stream<<std::endl;
68             for (std::size_t index = 0; index < shift; index++) {stream<<Prefix;}
69         }
70     }
71     return (((std::next(Paging.PagingTable.begin(), pageId))>SystemMemoryFrame_ID)*Frame.Size)+(virtualAddress-(((std::next(Paging.PagingTable.begin(), pageId))>ID)*Frame.Size));
72 }
73 #endif
74
75

```

.\ProcessMemoryPaging.hpp

```

1  #include <list>
2  #include <iostream>
3  #include "SystemMemoryFrame.hpp"
4  #ifndef ProcessMemoryPaging_hpp
5      #define ProcessMemoryPaging_hpp
6      #include "ProcessMemoryPagingTableData.hpp"
7      class ProcessMemoryPaging
8      {
9      private:
10     protected:
11     public:
12         std::size_t Number;
13         std::list<ProcessMemoryPagingTableData*> PagingTable;
14         SystemMemoryFrame& Frame;
15         ProcessMemoryPaging(ProcessMemoryPaging&);
16         ProcessMemoryPaging(std::size_t, SystemMemoryFrame&);
17         void deploy(std::size_t);
18         void print(char*, std::size_t, std::ostream&);
19     };
20 #endif
21 #ifndef ProcessMemoryPaging_cpp
22     #include "ProcessMemoryPaging.cpp"
23 #endif

```

.\ProcessMemoryPaging.cpp

```

1  #include "Debug.hpp"
2  #include <iomanip>
3  #include <cmath>
4  #ifndef ProcessMemoryPaging_hpp
5  #include "ProcessMemoryPaging.hpp"
6  #endif
7  #ifndef ProcessMemoryPaging_cpp
8  #define ProcessMemoryPaging_cpp
9  ProcessMemoryPaging::ProcessMemoryPaging(ProcessMemoryPaging& processMemoryPaging)
10 : Number(processMemoryPaging.Number), Frame(processMemoryPaging.Frame), PagingTable(processMemoryPaging.PagingTable)
11 {
12 }
13 ProcessMemoryPaging::ProcessMemoryPaging(std::size_t processMemorySize, SystemMemoryFrame& systemMemoryFrame)
14 : Number(processMemorySize/systemMemoryFrame.Size), Frame(systemMemoryFrame)
15 {
16     debug_Flag(std::clog);
17     for (std::size_t index = 0; index < Number; index++)
18     {
19         ProcessMemoryPagingTableData* buffer = new ProcessMemoryPagingTableData(index);
20         PagingTable.push_back(buffer);
21     }
22     debug_Flag(std::clog);
23 }
24 void ProcessMemoryPaging::deploy(std::size_t pageId)
25 {
26     (*std::next(PagingTable.begin(), pageId))->Valid = true;
27     (*std::next(PagingTable.begin(), pageId))->SystemMemoryFrame_ID = Frame.InvalidList.front()->ID;
28     Frame.InvalidList.front()->Valid = true;
29     Frame.InvalidList.pop_front();
30 }
31 void ProcessMemoryPaging::print(char* Prefix=(char*)"", std::size_t shift=0, std::ostream& stream=std::cout)
32 {
33     for (std::size_t index = 0; index < shift; index++) {stream<<Prefix;} stream<<"Process's Memory Paging Number: "<<Number<<std::endl;
34     for (std::size_t index = 0; index < shift; index++) {stream<<Prefix;} stream<<"Process's Memory Paging Size: "<<Frame.Size<<std::endl;
35     for (std::size_t index = 0; index < shift; index++) {stream<<Prefix;} stream<<"Process's Memory Paging Table: ";
36     for (std::size_t index = 0; index <= shift; index++) {stream<<Prefix;} stream<<"Page : [";
37     for (std::size_t index = 0; index < Number; index++) {stream<<std::setw(std::log10(Frame.Size))<<(*std::next(PagingTable.begin(), index))->ID <<" , ";}
38     stream<<"\b\b]"<<std::endl;
39     for (std::size_t index = 0; index <= shift; index++) {stream<<Prefix;} stream<<"Frame : [";
40     for (std::size_t index = 0; index < Number; index++) {stream<<std::setw(std::log10(Frame.Size))<<(*std::next(PagingTable.begin(), index))->SystemMemoryFrame_ID <<" , ";}
41     stream<<"\b\b]"<<std::endl;
42     for (std::size_t index = 0; index <= shift; index++) {stream<<Prefix;} stream<<"Valid : [";
43     for (std::size_t index = 0; index < Number; index++) {stream<<std::setw(std::log10(Frame.Size))<<(*std::next(PagingTable.begin(), index))->Valid <<" , ";}
44     stream<<"\b\b]"<<std::endl;
45 }
46 #endif

```

.\ProcessMemoryPagingTableData.hpp

```

1  #include <iostream>
2  #ifndef ProcessMemoryPagingTableData_hpp
3  #define ProcessMemoryPagingTableData_hpp
4  class ProcessMemoryPagingTableData
5  {
6  private:
7  protected:
8  public:
9      std::size_t ID;
10     std::size_t SystemMemoryFrame_ID;
11     bool Valid;
12     ProcessMemoryPagingTableData(std::size_t, bool, std::size_t);
13 };
14 #endif
15 #ifndef ProcessMemoryPagingTableData_cpp
16 #include "ProcessMemoryPagingTableData.cpp"
17 #endif

```

.\ProcessMemoryPagingTableData.cpp

```

1  #include "Debug.hpp"
2  #ifndef ProcessMemoryPagingTableData_hpp
3  #include "ProcessMemoryPagingTableData.hpp"
4  #endif
5  #ifndef ProcessMemoryPagingTableData_cpp
6  #define ProcessMemoryPagingTableData_cpp
7  ProcessMemoryPagingTableData::ProcessMemoryPagingTableData(std::size_t id, bool valid = false, std::size_t systemMemoryFrame_ID = std::numeric_limits<std::size_t>::max())
8 : ID(id), Valid(valid), SystemMemoryFrame_ID(systemMemoryFrame_ID)
9 {debug_Flag(std::clog);}
10 #endif

```

.\System.hpp

```

1  #include <list>
2  #include <string>
3  #ifndef System_hpp
4      #define System_hpp
5      #include "SystemMemory.hpp"
6      #include "Process.hpp"
7      class System
8      {
9      private:
10     protected:
11     public:
12         SystemMemory Memory;
13         std::size_t NumberDispatchRecord;
14         std::list<Process> Processes;
15         System(SystemMemory&);
16         void addProcess(std::string, std::size_t, std::size_t );
17         // std::size_t physicalAddress(std::size_t, std::size_t, bool, bool, char*, std::size_t, std::ostream&);
18         std::size_t physicalAddress(std::string, std::size_t, bool, bool, char*, std::size_t, std::ostream&);
19     };
20 #endif
21 #ifndef System_cpp
22     #include "System.cpp"
23 #endif

```

.\System.cpp

```

1  #include "Debug.hpp"
2  #include <string>
3  #include <exception>
4  #include "Container.hpp"
5  #define debug_Flag(stream) stream<<"Debug: \n { \n      File: \"\"<<__FILE__<<\"\", \n      Line: \"\"<<__LINE__<<\"\" }\"<<std::endl
6
7  #ifndef System_hpp
8      #include "System.hpp"
9  #endif
10 #ifndef System_cpp
11     #define System_cpp
12     System::System(SystemMemory& memory)
13         : Memory(memory), NumberDispatchRecord(0)
14     {}
15     void System::addProcess(std::string processName, std::size_t processMemorySize, std::size_t initialDeployNumber =0)
16     {
17         debug_Flag(std::clog);
18         try
19         {
20             // Processes.push_back((new Process(NumberDispatchRecord++, processName, processMemorySize, &Memory, initialDeployNumber)));
21             debug_Flag(std::clog);
22             auto* tmp = new Process(NumberDispatchRecord++, processName, processMemorySize, &Memory, initialDeployNumber );
23             debug_Flag(std::clog);
24             std::clog << tmp->Name<< std::endl;
25         }
26         catch(const std::exception& e)
27         {
28             std::cerr << e.what() << '\n';
29         }
30         debug_Flag(std::clog);
31     }
32     // std::size_t System::physicalAddress(std::size_t ProcessID , std::size_t virtualAddress, bool printCalculationProcess=false, bool newline=false, char* Prefix=(char*)"()", std::size_t shift=0, std::ostream& stream=std::clog)
33     // {
34     //     Container::select_pointer<std::list<Process>, Process, std::size_t>(Processes, ProcessID, [](Process& process, std::size_t& Process_ID) -> bool {return (process.ID==Process_ID);}) -> physicalAddress(virtualAddress, process.Memory.Frame.Size, printCalculationProcess, newline, Prefix, shift, stream);
35     // }
36     std::size_t System::physicalAddress(std::string ProcessName, std::size_t virtualAddress, bool printCalculationProcess=false, bool newline=false, char* Prefix=(char*)"()", std::size_t shift=0, std::ostream& stream=std::clog)
37     {
38         stream<<" Page: (" <<virtualAddress<<"/"<<Memory.Frame.Size<<") = "<<(virtualAddress/Memory.Frame.Size)<<std::endl;
39         stream<<" Physical Address: ";
40         return Container::select_pointer<std::list<Process>, Process, std::string>(Processes, ProcessName, [](Process& process, std::string& ProcessName) -> bool {return (process.Name==ProcessName);}) -> physicalAddress(virtualAddress, process.Memory.Frame.Size, printCalculationProcess, newline, Prefix, shift, stream);
41     }
42 #endif

```

.\SystemMemory.hpp

```
1  #include <list>
2  #ifndef SystemMemory_hpp
3      #define SystemMemory_hpp
4      #include "SystemMemoryFrame.hpp"
5      class SystemMemory
6      {
7      private:
8      protected:
9      public:
10         std::size_t Size;
11         SystemMemoryFrame Frame;
12         SystemMemory(SystemMemory&);
13         SystemMemory(std::size_t, std::size_t, bool);
14     };
15 #endif
16 #ifndef SystemMemory_cpp
17     #include "SystemMemory.cpp"
18 #endif
```

.\SystemMemory.cpp

```
1  #include "Debug.hpp"
2  #include <random>
3  #include <chrono>
4  #include <algorithm>
5  #ifndef SystemMemory_hpp
6      #include "SystemMemory.hpp"
7  #endif
8  #ifndef SystemMemory_cpp
9      #define SystemMemory_cpp
10     SystemMemory::SystemMemory(SystemMemory& systemMemory)
11         : Size(systemMemory.Size), Frame(systemMemory.Frame)
12     {
13         debug_Flag(std::clog);
14     }
15     SystemMemory::SystemMemory(std::size_t size, std::size_t frameSize, bool RandomFrameInvalidList=true)
16         : Size(size), Frame(Size, frameSize, RandomFrameInvalidList)
17     {
18         Size=size;
19         debug_Flag(std::clog);
20     }
21 #endif
```


.\SystemMemoryFrame.hpp

```

1  #include <list>
2  #include <map>
3  #ifndef SystemMemoryFrame_hpp
4      #define SystemMemoryFrame_hpp
5      #include "SystemMemoryFrameTableData.hpp"
6      class SystemMemoryFrame
7      {
8      private:
9      protected:
10     public:
11         std::size_t Size;
12         std::size_t Number;
13         std::list<SystemMemoryFrameTableData*> InvalidList;
14         // std::map<std::size_t, SystemMemoryFrameTableData*> FrameTable;
15         // std::list<SystemMemoryFrameTableData*> FrameTable;
16         SystemMemoryFrame(SystemMemoryFrame&);
17         SystemMemoryFrame(std::size_t , std::size_t, bool);
18     };
19 #endif
20 #ifndef SystemMemoryFrame_cpp
21     #include "SystemMemoryFrame.cpp"
22 #endif

```

.\SystemMemoryFrame.cpp

```

1  #include "Debug.hpp"
2  #include <random>
3  #include <chrono>
4  #include <algorithm>
5  #include "Processor.h"
6  #ifndef SystemMemoryFrame_hpp
7      #include "SystemMemoryFrame.hpp"
8  #endif
9  #ifndef SystemMemoryFrame_cpp
10     #define SystemMemoryFrame_cpp
11
12     SystemMemoryFrame::SystemMemoryFrame(SystemMemoryFrame& systemMemoryFrame)
13         : Size(systemMemoryFrame.Size), Number(systemMemoryFrame.Number), InvalidList(systemMemoryFrame.InvalidList)
14     {}
15     SystemMemoryFrame::SystemMemoryFrame(std::size_t systemMemorySize, std::size_t size, bool RandomInvalidList=true)
16         : Size(size), Number(systemMemorySize/Size)
17     {
18         debug_Flag(std::clog);
19         for (std::size_t index = 0; index < Number; index++)
20         {
21             SystemMemoryFrameTableData* buffer = new SystemMemoryFrameTableData(index);
22             InvalidList.push_back(buffer);
23             // FrameTable[buffer->ID]=*buffer;
24             // FrameTable.push_back(buffer);
25         }
26         if (RandomInvalidList)
27         { // std::random_shuffle(InvalidList.begin(), InvalidList.end());
28             /** @brief Random Number Generator */
29             std::mt19937 RandomNumberGenerator(std::chrono::system_clock::now().time_since_epoch().count());
30             /** @brief Random sort using insertion sort */
31             for (std::size_t index = 0; index < Number; index++)
32             /** @brief Randomly obtain one of the values of the unprocessed block to the current processing position */
33             Processor::exchange(*std::next(InvalidList.begin(), index), *std::next(InvalidList.begin(), (*(new std::uniform_int_distribution<size_t>(index, Number-1)))(RandomNumberGenerator))));
34         }
35         debug_Flag(std::clog);
36     }
37 #endif

```

.\SystemMemoryFrameTableData.hpp

```
1  #include <iostream>
2  #ifndef SystemMemoryFrameTableData_hpp
3      #define SystemMemoryFrameTableData_hpp
4      class SystemMemoryFrameTableData
5      {
6      private:
7      protected:
8      public:
9          const std::size_t ID;
10         bool Valid;
11         SystemMemoryFrameTableData(std::size_t, bool);
12     };
13 #endif
14 #ifndef SystemMemoryFrameTableData_cpp
15     #include "SystemMemoryFrameTableData.cpp"
16 #endif
```

.\SystemMemoryFrameTableData.cpp

```
1  #include "Debug.hpp"
2  #ifndef SystemMemoryFrameTableData_hpp
3      #include "SystemMemoryFrameTableData.hpp"
4  #endif
5  #ifndef SystemMemoryFrameTableData_cpp
6      #define SystemMemoryFrameTableData_cpp
7      SystemMemoryFrameTableData::SystemMemoryFrameTableData(std::size_t id, bool valid = false)
8          : ID(id), Valid(valid)
9      {
10         // debug_Flag(std::clog);
11     }
12 #endif
```

.\main.cpp

```

1  #include <iostream>
2  #include <iomanip>
3  #include <cmath>
4  #include <string>
5  #include <random>
6  #include <list>
7  #include <chrono>
8  #include "Debug.hpp"
9  #include "System.hpp"
10 // #include "Process.hpp"
11
12 template <typename Type> Type stdcin()
13 {
14     Type result;
15     std::cin>>result;
16     return result;
17 }
18
19 template <typename Type> Type stdcin(const char* InputHint, bool (*isInvalidValue)(Type), char* Prefix=(char*)"", std::size_t shift=0)
20 {
21     Type result;
22     bool InputError = true; while (InputError)
23     {
24         std::cout<<InputHint; std::cin>>result;
25         InputError = false;
26         if (!std::cin)
27         {
28             InputError = true;
29             std::cerr<<" ERROR: Unrecognized input."<<std::endl;
30             std::cin.clear();
31             std::cin.ignore(std::numeric_limits<std::streamsize>::max(), '\n');
32             continue;
33         }
34         if (isInvalidValue(result))
35         {
36             InputError = true;
37             std::cerr<<" ERROR: Invalid value detected: "<<result<<std::endl;
38             continue;
39         }
40     }
41     return result;
42 }
43
44 int main(int argc, char const *argv[])
45 {
46     System system(*(new SystemMemory(
47         stdcin<std::size_t>("Input System Memory Size: ", [](std::size_t buffer) -> bool {return (std::log2(buffer)-std::floor(std::log2(buffer))));},
48         stdcin<std::size_t>("Input System Memory Frame Size: ", [](std::size_t buffer) -> bool {return (std::log2(buffer)-std::floor(std::log2(buffer))));}
49     )));
50     debug_ObjectValue(std::clog, system.Memory.Size, "system.Memory.Size");\
51     std::size_t ProcessNumber = stdcin<std::size_t>("Input Number of Process: ", [](std::size_t buffer) -> bool {return false;});
52     for (std::size_t index = 0; index < ProcessNumber; index++)
53     {
54         std::cout<<" Information of Process["<<index<<"]: "<<std::endl;
55         system.addProcess(
56             stdcin<std::string>("Input Process Name: ", [](std::string buffer) -> bool {return false;}, (char*)" ", 2),
57             stdcin<std::size_t>("Input Process Memory Size: ", [](std::size_t buffer) -> bool {return false;}, (char*)" ", 2),
58             3
59         );
60         //debug_Flag(std::clog);
61         // debug_ObjectValue(std::clog, system.Processes.back().Name, "system.Processes.back()");
62         system.Processes.back().print((char*)" ", 2);
63     }
64     while (true)
65     {
66         system.physicalAddress(
67             stdcin<std::string>("Enter itinerary and Virtual Address: ", [](std::string buffer) -> bool {return false;}, (char*)" ", 2),
68             stdcin<std::size_t>("", [](std::size_t buffer) -> bool {return false;}, (char*)"", 0)
69         );
70     }
71
72     std::cout<<"main: Finish."<<std::endl;
73     return 0;
74 }

```

三、執行結果

```
Input System Memory Size: 1048576
Input System Memory Frame Size: 4096
Input Number of Process: 5
Information of Process[0]:
Input Process Name: p0
Input Process Memory Size: 10240
Process's Memory Paging Number: 3
Process's Memory Paging Size: 4096
Page : [ 0, 1, 2, 3]
Frame : [243, 199, 66, ]
Valid : [ 1, 1, 1, 0]
Information of Process[1]:
Input Process Name: p1
Input Process Memory Size: 30720
Process's Memory Paging Number: 8
Process's Memory Paging Size: 4096
Page : [ 0, 1, 2, 3, 4, 5, 6, 7, 8]
Frame : [220, 229, 52, , , , , , ]
Valid : [ 1, 1, 1, 0, 0, 0, 0, 0, 0]
Information of Process[2]:
Input Process Name: p2
Input Process Memory Size: 51200
Process's Memory Paging Number: 13
Process's Memory Paging Size: 4096
Page : [ 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13]
Frame : [125, 236, 66, , , , , , , , , , ]
Valid : [ 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
Information of Process[3]:
Input Process Name: p3
Input Process Memory Size: 71680
Process's Memory Paging Number: 18
Process's Memory Paging Size: 4096
Page : [ 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18]
Frame : [ 51, 205, 95, , , , , , , , , , , , , , ]
Valid : [ 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
Information of Process[4]:
Input Process Name: p4
Input Process Memory Size: 92160
Process's Memory Paging Number: 23
Process's Memory Paging Size: 4096
Page : [ 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23]
Frame : [ 98, 8, 32, , , , , , , , , , , , , , , , , , , ]
Valid : [ 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
```

```

Enter itinerary and Virtual Address: p0 886
Information of Process[0]:
  Input Process Name: p0
  Input Process Memory Size: 10240
  Process's Memory Paging Number: 3
  Process's Memory Paging Size: 4096
  Page : [ 0, 1, 2, 3]
  Frame : [243, 199, 66, ]
  Valid : [ 1, 1, 1, 0]
  Page: floor(886/4096) = 0
  Physical Address: (243*4096)+(10240%4096) = 997376
Enter itinerary and Virtual Address: p1 13299
Information of Process[1]:
  Input Process Name: p1
  Input Process Memory Size: 30720
  Process's Memory Paging Number: 8
  Process's Memory Paging Size: 4096
  Page : [ 0, 1, 2, 3, 4, 5, 6, 7, 8]
  Frame : [220, 229, 52, 188, , , , , ]
  Valid : [ 1, 1, 1, 0, 0, 0, 0, 0, 0]
  Page: floor(13299/4096) = 3
  Physical Address: (188*4096)+(13299%4096) = 771059
Enter itinerary and Virtual Address: p2 22354
Information of Process[2]:
  Input Process Name: p2
  Input Process Memory Size: 51200
  Process's Memory Paging Number: 13
  Process's Memory Paging Size: 4096
  Page : [ 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13]
  Frame : [125, 236, 66, , 221, , , , , , , , ]
  Valid : [ 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
  Page: floor(22356/4096) = 5
  Physical Address: (221*4096)+(22356%4096) = 907092
Enter itinerary and Virtual Address: p3 46858
Information of Process[3]:
  Input Process Name: p3
  Input Process Memory Size: 71680
  Process's Memory Paging Number: 18
  Process's Memory Paging Size: 4096
  Page : [ 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18]
  Frame : [ 51, 205, 95, , , , , , , , , 207, , , , , , , ]
  Valid : [ 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
  Page: floor(46858/4096) = 11
  Physical Address: (207*4096)+(46858%4096) = 849674
Enter itinerary and Virtual Address: p4 81127
Information of Process[4]:
  Input Process Name: p4
  Input Process Memory Size: 92160
  Process's Memory Paging Number: 23
  Process's Memory Paging Size: 4096
  Page : [ 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23]
  Frame : [ 98, 8, 32, , , , , , , , , , , , , , , , 202, , , , ]
  Valid : [ 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
  Page: floor(81127/4096) = 19
  Physical Address: (202*4096)+(81127%4096) = 830695
Enter itinerary and Virtual Address: ^C

```