Assignment 3

Topic: Memory Allocation Techniques

Theory

What is Dynamic Partitioning?

Unlike Fixed partitioning, the partitions here are of variable length. Due to this, each process is given exactly as much memory is required and no more.
This leads to the formation of holes in the memory, AKA **External Fragmentation.** Hence, we must use compaction to shift processes so that they are all in one block and all free memory is in one block.

The Operating System decides which free block is to be allocated to which process. For this, Operating System uses various Partitioning Placement algorithms such as:
Best Fit, Next Fit, Worst Fit, First Fit

What is First Fit?

This algorithm scans the memory locations from the beginning and chooses the 1st available block that has sufficient enough memory for the process and allocates it to the process. This is one of the fastest algorithms but this leads to having many processes loaded in the front end of memory that must be searched over when trying to find a free block.

What is Best Fit?

This algorithm scans the full memory to find a free memory block that is closest to the required memory size. This causes poor performance because this algorithm has to scan the full memory always unlike First Fit. This also leads to a lot of very small holes in the memory and hence memory compaction has to be done more often.

What is Worst Fit?

This algorithm also scans the full memory to find the largest free memory block possible and allocates the process, its required memory there, so that it leaves a usable memory fragment. This also causes poor performance because just like Best Fit, it has to scan the full memory always.

Dheeraj Lalwani
1902085

Example

Assuming that following is the current status of the memory:

P1 45kb	P2 35kb	free 40kb	P3 60kb	free 50kb	P4 25kb	free 35kb

And Suppose, we have a process **P5** which requires **35 kb** of memory.

First Fit: Example

First Fit will choose the 3rd section which has **40kb** of **free** memory and allocate it to **P5**, leaving behind a fragment of 40kb - 35kb = 5 kb fragment. Following will be the state of the memory after insertion using First Fit.

P1 45kb	P2 35kb	P5 35kb	free 5kb	P3 60kb	free 5 kb	P4 25kb	free 35kb

Best Fit: Example

Best Fit will choose the 7th section which has **35kb** of **free** memory and allocate it to **P5**, leaving behind a fragment of 35kb - 35kb = 0 kb fragment. Following will be the state of the memory after insertion using Best Fit.

P1 45kb	P2 35kb	free 40kb	P3 60kb	free 50kb	P4 25kb	P5 35kb

Worst Fit: Example

First Fit will choose the 5th section which has **50kb** of **free** memory and allocate it to **P5**, leaving behind a fragment of 50kb - 35kb = 15 kb fragment. Following will be the state of the memory after insertion using Worst Fit.

P1 45kb	P2 35kb	free 40kb	P3 60kb	P5 35kb	free 15kb	P4 25kb	free 35kb

Code

```
#include <stdio.h>
#include <stdbool.h>
#include <stdlib.h>
#include <ctype.h>
#include <string.h>

struct section
{
      int memory;
      bool occupiedStatus;
      char process[5];
      struct section *next;
};

struct memory
{
      int sectionCount;
      struct section *start;
      struct section *end;
};

struct section *sectionStart, *sectionEnd, *sectionNext, *sectionNew, *sectionPrevious,
*largestFreeBlock, *largestFreeBlockPrevious, *smallestFreeBlock,
*smallestFreeBlockPrevious;

struct memory *memInfo;

void initializeMemory()
{
      printf("Initializing memory...\n");
      memInfo = (struct memory *)malloc(sizeof(struct memory));
      memInfo->sectionCount = 0;
      memInfo->start = NULL;
      memInfo->end = NULL;
}

void inputMemoryStatus()
{
      bool ongoing = true;
      char yesNo;
      printf("This will take input of the current memory status.\n");

      do
      {
            if (memInfo->start == NULL)
            {
                  // struct section *sectionStart;
                  sectionNew = (struct section *)malloc(sizeof(struct section));
                  printf("Please enter Process name (Enter free if the memory block is
free) : ");
                  scanf("%s", sectionNew->process);
                  getchar();
                  printf("Please enter %s memory : ", sectionNew->process);
                  scanf("%d", &sectionNew->memory);
                  getchar();
                  if (strcmp(sectionNew->process, "free") == 0)
                  {
                        sectionNew->occupiedStatus = false;
                  }
                  else
                  {
                        sectionNew->occupiedStatus = true;
                  }
                  printf("Do you want to add next section [Y/n] : ");
                  scanf("%c", &yesNo);
                  if (tolower(yesNo) == 'n')
                  {
                        ongoing = false;
```

```
                        sectionNew->next = NULL;
                        memInfo->start = sectionNew;
                        memInfo->end = sectionNew;
                        break;
                  }
                  else
                  {
                        memInfo->start = sectionNew;
                        memInfo->end = sectionNew;
                        continue;
                  }
            }
            else
            {
                  sectionNew = (struct section *)malloc(sizeof(struct section));
                  printf("Please enter Process name (Enter free if the memory block is
free) : ");
                  scanf("%s", sectionNew->process);
                  getchar();
                  printf("Please enter %s memory : ", sectionNew->process);
                  scanf("%d", &sectionNew->memory);
                  getchar();
                  if (strcmp(sectionNew->process, "free") == 0)
                  {
                        sectionNew->occupiedStatus = false;
                  }
                  else
                  {
                        sectionNew->occupiedStatus = true;
                  }
                  printf("Do you want to add next section [Y/n] : ");
                  scanf("%c", &yesNo);
                  if (tolower(yesNo) == 'n')
                  {
                        ongoing = false;
                        sectionNew->next = NULL;
                        memInfo->end->next = sectionNew;
                        memInfo->end = sectionNew;
                        break;
                  }
                  else
                  {
                        sectionNew->next = NULL;
                        memInfo->end->next = sectionNew;
                        memInfo->end = sectionNew;
                        continue;
                  }
            }
      } while (ongoing);
}

void inputNewProcess()
{
      sectionNew = (struct section *)malloc(sizeof(struct section));
      if (sectionNew != NULL)
      {
            printf("Please enter Process name : ");
            scanf("%s", sectionNew->process);
            getchar();
            printf("Please enter %s memory : ", sectionNew->process);
            scanf("%d", &sectionNew->memory);
            getchar();
            sectionNew->occupiedStatus = true;
            sectionNew->next = NULL;
      }
      else
      {
            printf("Insufficient memory, exiting .... \n");
      }
}
```

```
void printMemoryStatus()
{
	char status[3];
	printf("This will print the status of the memory.\n");
	printf("|  %8s  |  %7s  |  %7s  |\n", "Section", "Memory", "Status");
	sectionNext = memInfo->start;
	while (sectionNext != NULL)
	{
		if (sectionNext->occupiedStatus == 0)
		{
			strcpy(status, "F");
		}
		else
		{
			strcpy(status, "NF");
		}
		printf("|  %8s  |  %7d  |  %7s  |\n", sectionNext->process,
sectionNext->memory, status);
		sectionNext = sectionNext->next;
	}
	printf("=============================================\n");
}

void insertWorstFit()
{
	int highestMemory = 0;
	printf("This will insert the new process in the memory, using worst fit.\n");
	sectionPrevious = NULL;
	sectionNext = memInfo->start;
	largestFreeBlock = NULL;
	largestFreeBlockPrevious = NULL;
	while (sectionNext != NULL)
	{
		if (sectionNext->occupiedStatus == 0 && sectionNext->memory > highestMemory &&
sectionNext->memory >= sectionNew->memory)
		{
			highestMemory = sectionNext->memory;
			largestFreeBlock = sectionNext;
			largestFreeBlockPrevious = sectionPrevious;
		}
		sectionPrevious = sectionNext;
		sectionNext = sectionNext->next;
	}
	if (largestFreeBlock == NULL)
	{
		printf("No free blocks available... Exiting...\n");
		exit(0);
	}
	else if (largestFreeBlock != NULL && largestFreeBlockPrevious == NULL &&
largestFreeBlock->memory > sectionNew->memory)
	{
		sectionNew->next = memInfo->start;
		memInfo->start->memory = memInfo->start->memory - sectionNew->memory;
		memInfo->start = sectionNew;
	}
	else if (largestFreeBlock != NULL && largestFreeBlockPrevious == NULL &&
largestFreeBlock->memory == sectionNew->memory)
	{
		sectionNew->next = memInfo->start->next;
		memInfo->start = sectionNew;
	}

	else if (largestFreeBlock != NULL && largestFreeBlockPrevious != NULL &&
largestFreeBlock->memory > sectionNew->memory)
	{
		sectionNew->next = largestFreeBlockPrevious->next;
		largestFreeBlock->memory = largestFreeBlock->memory - sectionNew->memory;
		largestFreeBlockPrevious->next = sectionNew;
	}
	else if (largestFreeBlock != NULL && largestFreeBlockPrevious != NULL &&
largestFreeBlock->memory == sectionNew->memory)
```

```
	{
		sectionNew->next = largestFreeBlockPrevious->next->next;
		largestFreeBlockPrevious->next = sectionNew;
	}
	else
	{
		printf("No sufficient memory slot available. Exiting...\n");
	}
	printMemoryStatus();
}

void insertFirstFit()
{
	printf("This will insert the new process in the memory, using first fit.\n");
	sectionPrevious = NULL;
	sectionNext = memInfo->start;
	while (sectionNext != NULL)
	{
		if (sectionNext->occupiedStatus == 0 && sectionNext->memory >=
sectionNew->memory)
		{
			break;
		}
		sectionPrevious = sectionNext;
		sectionNext = sectionNext->next;
	}
	if (sectionPrevious == NULL && sectionNext->occupiedStatus == 0 &&
sectionNext->memory > sectionNew->memory)
	{
		sectionNew->next = memInfo->start;
		memInfo->start->memory = memInfo->start->memory - sectionNew->memory;
		memInfo->start = sectionNew;
	}
	else if (sectionPrevious == NULL && sectionNext->occupiedStatus == 0 &&
sectionNext->memory == sectionNew->memory)
	{
		sectionNew->next = memInfo->start->next;
		memInfo->start = sectionNew;
	}
	else if (sectionPrevious != NULL && sectionNext->occupiedStatus == 0 &&
sectionNext->memory > sectionNew->memory)
	{
		sectionNew->next = sectionPrevious->next;
		sectionNext->memory = sectionNext->memory - sectionNew->memory;
		sectionPrevious->next = sectionNew;
	}
	else if (sectionPrevious != NULL && sectionNext->occupiedStatus == 0 &&
sectionNext->memory == sectionNew->memory)
	{
		sectionNew->next = sectionPrevious->next->next;
		sectionPrevious->next = sectionNew;
	}
	else
	{
		printf("No sufficient memory slot available. Exiting...\n");
	}
	printMemoryStatus();
}

void insertBestFit()
{
	// DOING
	printf("This will insert the new process in the memory, using best fit.\n");
	sectionNext = memInfo->start;
	while (sectionNext->occupiedStatus != 0)
	{
		sectionNext = sectionNext->next;
	}
	int lowestMemory = sectionNext->memory;
	sectionPrevious = NULL;
	sectionNext = memInfo->start;
```

```
	smallestFreeBlock = NULL;
	smallestFreeBlockPrevious = NULL;
	while (sectionNext != NULL)
	{
		if (sectionNext->occupiedStatus == 0 && sectionNext->memory < lowestMemory &&
sectionNext->memory >= sectionNew->memory)
		{
			lowestMemory = sectionNext->memory;
			smallestFreeBlock = sectionNext;
			smallestFreeBlockPrevious = sectionPrevious;
		}
		sectionPrevious = sectionNext;
		sectionNext = sectionNext->next;
	}
	if (smallestFreeBlock == NULL)
	{
		printf("No free blocks available... Exiting...\n");
		exit(0);
	}
	else if (smallestFreeBlock != NULL && smallestFreeBlockPrevious == NULL &&
smallestFreeBlock->memory > sectionNew->memory)
	{
		sectionNew->next = memInfo->start;
		memInfo->start->memory = memInfo->start->memory - sectionNew->memory;
		memInfo->start = sectionNew;
	}
	else if (smallestFreeBlock != NULL && smallestFreeBlockPrevious == NULL &&
smallestFreeBlock->memory == sectionNew->memory)
	{
		sectionNew->next = memInfo->start->next;
		memInfo->start = sectionNew;
	}

	else if (smallestFreeBlock != NULL && smallestFreeBlockPrevious != NULL &&
smallestFreeBlock->memory > sectionNew->memory)
	{
		sectionNew->next = smallestFreeBlockPrevious->next;
		smallestFreeBlock->memory = smallestFreeBlock->memory - sectionNew->memory;
		smallestFreeBlockPrevious->next = sectionNew;
	}
	else if (smallestFreeBlock != NULL && smallestFreeBlockPrevious != NULL &&
smallestFreeBlock->memory == sectionNew->memory)
	{
		sectionNew->next = smallestFreeBlockPrevious->next->next;
		smallestFreeBlockPrevious->next = sectionNew;
	}
	else
	{
		printf("No sufficient memory slot available. Exiting...\n");
	}
	printMemoryStatus();
}

int menu()
{
	int choice;
	printf("1) Insert With Best Fit\n");
	printf("2) Insert With First Fit\n");
	printf("3) Insert With Worst Fit\n");
	printf("4) Exit\n");
	printf("Please enter your choice: ");
	scanf("%d", &choice);
	return choice;
}

int main()
{
	int choice;
	initializeMemory();
	printf("Welcome to memory simulation system.\n\n");
	inputMemoryStatus();
```

```
	printMemoryStatus();
	inputNewProcess();
	do
	{
		choice = menu();
		if (choice == 1)
		{
			insertBestFit();
		}
		else if (choice == 2)
		{
			insertFirstFit();
		}
		else if (choice == 3)
		{
			insertWorstFit();
		}
		else
		{
			exit(0);
		}
	} while (choice <= 4 && choice >= 1);
	return 0;
}
```

Output

User given input

```
D:\Dheeraj\Study\SEM 4\OperatingSystemPracticals\MemoryAllocationTechniques>AllocMemory
Initializing memory...
Welcome to memory simulation system.

This will take input of the current memory status.
Please enter Process name (Enter free if the memory block is free) : P1
Please enter P1 memory : 100
Do you want to add next section [Y/n] : y
Please enter Process name (Enter free if the memory block is free) : free
Please enter free memory : 300
Do you want to add next section [Y/n] : y
Please enter Process name (Enter free if the memory block is free) : P2
Please enter P2 memory : 40
Do you want to add next section [Y/n] : y
Please enter Process name (Enter free if the memory block is free) : free
Please enter free memory : 50
Do you want to add next section [Y/n] : y
Please enter Process name (Enter free if the memory block is free) : P3
Please enter P3 memory : 150
Do you want to add next section [Y/n] : y
Please enter Process name (Enter free if the memory block is free) : free
Please enter free memory : 240
Do you want to add next section [Y/n] : y
Please enter Process name (Enter free if the memory block is free) : p4
Please enter p4 memory : 200
Do you want to add next section [Y/n] : y
Please enter Process name (Enter free if the memory block is free) : free
Please enter free memory : 400
Do you want to add next section [Y/n] : n
```

```
This will print the status of the memory.
|   Section   |   Memory   |   Status   |
|         P1  |        100 |         NF |
|       free  |        300 |          F |
|         P2  |         40 |         NF |
|       free  |         50 |          F |
|         P3  |        150 |         NF |
|       free  |        240 |          F |
|         p4  |        200 |         NF |
|       free  |        400 |          F |
=============================================
```

Insert Using First Fit

```
Please enter Process name : P5
Please enter P5 memory : 200
1) Insert With Best Fit
2) Insert With First Fit
3) Insert With Worst Fit
4) Exit
Please enter your choice: 2
This will insert the new process in the memory, using first fit.
This will print the status of the memory.
|   Section  |   Memory  |   Status  |
|        P1  |      100  |       NF  |
|        P5  |      200  |       NF  |
|      free  |      100  |        F  |
|        P2  |       40  |       NF  |
|      free  |       50  |        F  |
|        P3  |      150  |       NF  |
|      free  |      240  |        F  |
|        p4  |      200  |       NF  |
|      free  |      400  |        F  |
=============================================
```

Dheeraj Lalwani
1902085

08/02/2021

Insert Using Best Fit

```
Please enter Process name : P5
Please enter P5 memory : 200
1) Insert With Best Fit
2) Insert With First Fit
3) Insert With Worst Fit
4) Exit
Please enter your choice: 1
This will insert the new process in the memory, using best fit.
This will print the status of the memory.
|   Section  |   Memory  |   Status  |
|        P1  |      100  |       NF  |
|      free  |      300  |        F  |
|        P2  |       40  |       NF  |
|      free  |       50  |        F  |
|        P3  |      150  |       NF  |
|        P5  |      200  |       NF  |
|      free  |       40  |        F  |
|        P4  |      200  |       NF  |
|      free  |      400  |        F  |
=============================================
```

Insert Using Worst Fit

```
Please enter Process name : P5
Please enter P5 memory : 200
1) Insert With Best Fit
2) Insert With First Fit
3) Insert With Worst Fit
4) Exit
Please enter your choice: 3
This will insert the new process in the memory, using worst fit.
This will print the status of the memory.
|   Section  |   Memory  |   Status  |
|        P1  |      100  |       NF  |
|      free  |      300  |        F  |
|        P2  |       40  |       NF  |
|      free  |       50  |        F  |
|        P3  |      150  |       NF  |
|      free  |      240  |        F  |
|        P4  |      200  |       NF  |
|        P5  |      200  |       NF  |
|      free  |      200  |        F  |
=============================================
```

Dheeraj Lalwani
1902085

08/02/2021