Script started on Mon Mar 4 20:29:39 2019
praveen@praveen$ cat Pag#ing.c

```
#include<stdio.h>
#include<stdlib.h>
int pgsize, memsize;
int nframes, nfree;
int frames[100];
typedef struct pagetable {
        int pageno;
        int frameno;
        struct pagetable* next;
}Pagetable;
Pagetable* process[10];
typedef struct freeframe {
        int frameno;
        struct freeframe* next;
}Freeframe;
Freeframe* fhead;
void printtable(Pagetable *head)
{
  for(Pagetable *temp = head; temp != NULL; temp = temp->next)
    printf("Page %d: Frame %d\n", temp->pageno, temp->frameno);
}
void allocate()
{
        int p, m, np;
        Pagetable *head, *temp, *prev;
        printf("Process ID: ");
        scanf("%d", &p);
        printf("Memory required in KB: ");
        scanf("%d", &m);
        np = m / pgsize;
        if(m*1.0/pgsize > np) np++;
        printf("No. of pages = %d\n", np);
        if(np > nfree) {
                printf("Not available!\n");
                return;
        }
        for(int i = 0; i < np; i++)
        {
                temp = (Pagetable*)malloc(sizeof(Pagetable));
                temp->pageno = i;
                temp->frameno = fhead->frameno;
                temp->next = NULL;
                Freeframe* t = fhead;
                fhead = fhead->next;


                free(t);
                nfree--;
                if(i == 0) head = temp;
```

```
                else prev->next = temp;
                prev = temp;
        }
        process[p] = head;
        printtable(process[p]);
}
void deallocate()
{
        int p;
        printf("PID: ");
        scanf("%d", &p);
        Pagetable *pd, *t, *d;
        int f;
        Freeframe *pf, *tf;
   pf = fhead;
        if(fhead != NULL) for(; pf->next != NULL; pf = pf->next);
        pd = process[p];
        if(process[p] == NULL) {
                printf("Not allocated!\n");
                return;
        }
        for(t = pd; t != NULL;)
        {
                d = t;
                f = d->frameno;
                tf = (Freeframe*)malloc(sizeof(Freeframe));
                tf->frameno = f;
                tf->next = NULL;
      if(pf != NULL) {
        pf->next = tf;
        pf = tf;
      }
      else {
        pf = tf;
        fhead = pf;
      }
                t = t->next;
      nfree++;
                free(d);
        }
        process[p] = NULL;
        printf("Deallocated!\n");
}
void showtables()
{
        for(int i = 0; i < 10; i++) {
      if(process[i] != NULL) printf("PID: %d\n", i);
      printtable(process[i]);
                if(process[i] != NULL) printf("\n");
        }
}
```

```
void addressmap()
{
        int p, la, pn, fn, o, pa;
        printf("PID: ");
        scanf("%d", &p);
        printf("Logical address: ");
        scanf("%d", &la);
        pn = la / pgsize;
        o = la % pgsize;
        Pagetable *h = process[p], *t;
        t = h;
        for(int i = 0; i < pn; i++)
        {
                t = t->next;
        }
        fn = t->frameno;
        pa = fn * pgsize + o;
        printf("Physical address: %d\n", pa);
}
void showfree()
{
  for(Freeframe *temp = fhead; temp != NULL; temp = temp->next)
    printf("%d ", temp->frameno);
  printf("\n");
}
int main()
{
        Freeframe *temp, *prev;
        printf("Total physical memory in KB: ");
        scanf("%d", &memsize);
        printf("Page size in KB: ");
        scanf("%d", &pgsize);
        nframes = memsize / pgsize;
        printf("No. of frames = %d\n", nframes);
        for(int i = 0; i < nframes/2; i++) {
                int fr = random()%nframes;
      if(frames[fr] != 1) {
        frames[fr] = 1;
        nfree++;
        temp = (Freeframe*)malloc(sizeof(Freeframe));
        temp->frameno = fr;
        temp->next = NULL;
        if(nfree == 1) fhead = temp;
        else prev->next = temp;
        prev = temp;
      }
      else i--;
        }
        int ch = 0;
        do {
```

```
		printf("1. Process Request 2. Deallocate 3. Show all Page Tables 4. Show all Free Frames 5. Address Mapping 6. Exit\nChoice: ");
		scanf("%d", &ch);
		switch(ch) {
			case 1: allocate(); break;
			case 2: deallocate(); break;
			case 3: showtables(); break;
			case 4: showfree(); break;
			case 5: addressmap(); break;
			case 6: exit(0);
		}
	}while(1);
	return 0;
}
praveen@praveen$ gcc P#aging.c -o P#aging
praveen@praveen$ ./P#aging
Total physical memory in KB: 20
Page size in KB: 2
No. of frames = 10
1. Process Request 2. Deallocate 3. Show all Page Tables 4. Show all Free Frames 5. Address Mapping 6. Exit
Choice: 1
Process ID: 1
Memory required in KB: 5
No. of pages = 3
Page 0: Frame 3
Page 1: Frame 6
Page 2: Frame 7
1. Process Request 2. Deallocate 3. Show all Page Tables 4. Show all Free Frames 5. Address Mapping 6. Exit
Choice: 4
5 2
1. Process Request 2. Deallocate 3. Show all Page Tables 4. Show all Free Frames 5. Address Mapping 6. Exit
Choice: 1
Process ID: 2
Memory required in KB: 4
No. of pages = 2
Page 0: Frame 5
Page 1: Frame 2
1. Process Request 2. Deallocate 3. Show all Page Tables 4. Show all Free Frames 5. Address Mapping 6. Exit
Choice: 4

1. Process Request 2. Deallocate 3. Show all Page Tables 4. Show all Free Frames 5. Address Mapping 6. Exit
Choice: 2
PID: 2
Deallocated!
1. Process Request 2. Deallocate 3. Show all Page Tables 4. Show all Free Frames 5. Address Mapping 6. Exit
```

Choice: 4
5 2
1. Process Request 2. Deallocate 3. Show all Page Tables 4. Show all Free Frames 5. Address Mapping 6. Exit
Choice: 1
Process ID: 3
Memory required in KB: 1
No. of pages = 1
Page 0: Frame 5
1. Process Request 2. Deallocate 3. Show all Page Tables 4. Show all Free Frames 5. Address Mapping 6. Exit
Choice: 3
PID: 1
Page 0: Frame 3
Page 1: Frame 6
Page 2: Frame 7

PID: 3
Page 0: Frame 5

1. Process Request 2. Deallocate 3. Show all Page Tables 4. Show all Free Frames 5. Address Mapping 6. Exit
Choice: 4
2
1. Process Request 2. Deallocate 3. Show all Page Tables 4. Show all Free Frames 5. Address Mapping 6. Exit
Choice: 5
PID: 1
Logical address: 2048
Physical address: 12288
1. Process Request 2. Deallocate 3. Show all Page Tables 4. Show all Free Frames 5. Address Mapping 6. Exit
Choice: 6
praveen@praveen$ exit
exit

Script done on Mon Mar 4 20:31:24 2019