**Fork():**

#include<stdio.h>

#include<unistd.h>

int main()

{

pid\_t pid,mypid,myppid;

pid=getpid();

pid=fork();

if(pid<0)

{

perror("fork() failure ");

return 1;

}

if(pid==0)

{

printf("\ni am in child porcess");

mypid=getpid();

myppid=getppid();

printf("\nprocess id =%d parent id=%d",mypid,myppid);

}

else

{

sleep(2);

printf("\ni am in parent porcess");

mypid=getpid();

myppid=getppid();

printf("\nprocess id =%d parent id=%d",mypid,myppid);

printf("\nnewly create child process=%d",pid);

}

return 0;

}

**Nice():**

#include<stdio.h>

#include<unistd.h>

int main()

{

pid\_t pid,retnice;

int i;

printf(" \npress del to stop to process \n ");

pid=fork();

for(i=0;i<4;i++)

{

if(pid==0)

{

retnice=nice(-4);

printf("child get higher cpu priority=%d\n",retnice);

sleep(1);

}

else

{

retnice=nice(5);

printf("parent get lower cpu priority=%d\n",retnice);

}

}

}

**Orphan():**

#include<stdio.h>

#include<unistd.h>

int main()

{

int pid=fork();

if(pid>0)

{

printf("parent process\n");

printf("id:%d\n",getpid());

}

if(pid==0)

{

printf("\nchild process\n");

printf("id:%d\n",getpid());

printf("parent id:%d",getppid());

sleep(10);

}

else

{

printf("failed to create child process");

}

return 0;

}

**FIFO:**

#include<stdio.h>

int main() {

int fsize, page\_faults = 0, index = 0;

int ref[] = {3, 4, 5, 6, 3, 4, 7, 3, 4, 5, 6, 7, 2, 4, 6};

int n = sizeof(ref) / sizeof(ref[0]);

printf("Enter number of frames:");

scanf("%d",&fsize);

int frame[fsize];

for (int i = 0; i < fsize; i++)

frame[i] = -1;

for (int i = 0; i < n; i++)

{

int found = 0;

for (int j = 0; j < fsize; j++)

if (frame[j] == ref[i])

found = 1;

if (!found)

{

frame[index] = ref[i];

index = (index + 1) % fsize;

page\_faults++;

}

printf("%d: ",ref[i]);

for (int j = 0; j < fsize; j++)

printf("%d ", frame[j] == -1 ? -1 : frame[j]);

printf("\n");

}

printf("Total Page Faults: %d\n", page\_faults);

return 0;

}

**LRU:**

#include <stdio.h>

int search(int frames[], int n, int page) {

for (int i = 0; i < n; i++)

if (frames[i] == page)

return i;

return -1;

}

void LRU(int pages[], int n, int frames[], int fsize) {

int time[100], count = 0, faults = 0;

for (int i = 0; i < fsize; i++)

frames[i] = -1;

for (int i = 0; i < n; i++) {

int pos = search(frames, fsize, pages[i]);

if (pos == -1)

{

int lru = 0;

if (count < fsize)

pos = count++;

else {

for (int j = 1; j < fsize; j++)

if (time[j] < time[lru])

lru = j;

pos = lru;

}

frames[pos] = pages[i];

faults++;

}

time[pos] = i;

printf("\nFrames: ");

for (int j = 0; j < fsize; j++)

if (frames[j] != -1)

printf("%d ", frames[j]);

else printf("- ");

}

printf("\nTotal Page Faults: %d\n", faults);

}

int main()

{

int ref[] = {3, 4, 5, 4, 3, 4, 7, 2, 4, 5, 6, 7, 2, 4, 6};

int fsize ;

printf("enter frames size");

scanf("%d",&fsize);

int n = sizeof(ref) / sizeof(ref[0]);

int frames[fsize];

LRU(ref, n, frames, fsize);

return 0;

}

**OPT:**

#include <stdio.h>

int predict(int ref[], int n, int frames[], int fsize, int idx)

{

for (int i = 0; i < fsize; i++)

{

int j;

for (j = idx; j < n; j++)

if (frames[i] == ref[j])

break;

if (j == n)

return i;

}

return 0;

}

void optimal(int ref[], int n,int frames[], int fsize)

{

int faults = 0, count = 0;

for (int i = 0; i < fsize; i++)

frames[i] = -1;

for (int i = 0; i < n; i++)

{

int found = 0;

for (int j = 0; j < fsize; j++)

if (frames[j] == ref[i])

{

found = 1;

break;

}

if (!found)

{

if (count < fsize) frames[count++] = ref[i];

else frames[predict(ref, n, frames, fsize, i)] = ref[i];

faults++;

}

for (int j = 0; j < fsize; j++)

printf(frames[j] == -1 ? "- " : "%d ", frames[j]);

printf("\n");

}

printf("Page Faults: %d\n", faults);

}

int main() {

int ref[] = {8, 5, 7, 8, 5, 7, 2, 3, 7, 3, 5, 9, 4, 6, 2};

int fsize ;

printf("enter frames size");

scanf("%d",&fsize);

int n = sizeof(ref) / sizeof(ref[0]);

int frames[fsize];

optimal(ref, n, frames, fsize);

return 0;

}

**Bubble, insertion sort in 24 (A)slips:**

#include <stdio.h>

#include <unistd.h>

#include <sys/wait.h>

void bubble\_sort(int arr[], int n) {

for (int i = 0; i < n-1; i++)

for (int j = 0; j < n-i-1; j++)

if (arr[j] > arr[j+1])

{

int t = arr[j];

arr[j] = arr[j+1];

arr[j+1] = t;

}

}

void insertion\_sort(int arr[], int n)

{

for (int i = 1; i < n; i++)

{

int key = arr[i], j = i - 1;

while (j >= 0 && arr[j] > key)

arr[j+1] = arr[j], j--;

arr[j+1] = key;

}

}

int main() {

int arr[10], n;

printf("Enter n: ");

scanf("%d", &n);

for (int i = 0; i < n; i++)

scanf("%d", &arr[i]);

pid\_t pid = fork();

if (pid == 0) {

insertion\_sort(arr, n);

printf("Child (Insertion Sort): ");

} else {

wait(NULL);

bubble\_sort(arr, n);

printf("Parent (Bubble Sort): ");

}

for (int i = 0; i < n; i++)

printf("%d ", arr[i]);

printf("\n");

return 0;

}