#include <stdio.h>

#include <stdlib.h>

#include <sys/ipc.h>

#include <sys/shm.h>

#include <sys/wait.h>

#include <unistd.h>

#include <string.h>

#define BUFFER\_SIZE 10

#define MAX\_LEN 256

#define KEY\_NUM 1004

#define NUM\_INPUT 100

#define TRUE 1

#define FALSE 0

//shared memory structure

typedef struct{

char data[BUFFER\_SIZE][MAX\_LEN];

int in;

int out;

}Buffer;

void producer(Buffer \* buffer);

void consumer(Buffer \* buffer);

int main(){

int pid;

int shm\_id;

Buffer \* buffer;

Buffer \* buffer\_parent;

Buffer \* buffer\_child;

//make shared memory

shm\_id = shmget(KEY\_NUM, sizeof(Buffer), IPC\_CREAT|0666);

if(shm\_id == -1){

printf("Failed to create shared memory\n");

return -1;

}

//attach shared memory to initialize shared memory

buffer = shmat(shm\_id, (void\*)0, 0);

if(buffer == (void\*)-1){

printf("Faied to attach shared memory\n");

}

//initialize shared memory

else{

buffer->in = 0;

buffer->out = 0;

if(shmdt((void\*)buffer) == -1)

printf("Failed to detach shared memory\n");

}

//make child process that execute consumer

pid = fork();

if(pid == 0){

printf("\nChild buffer is operating\n");

//attach child process to shared memory

buffer\_child = shmat(shm\_id, (void\*)0, 0);

if(buffer\_child == (void\*)-1){

printf("Failed to attach shared memory of consumer\n");

return -1;

}

//execute consumer function

consumer(buffer\_child);

//detach child buffer from shared memory

if(shmdt((void\*)buffer\_child) == -1)

printf("Failed to detach shared memory of producer\n");

}

//parent process execute producer;

else{

//attach parent process to shared memory

buffer\_parent = shmat(shm\_id, (void\*)0, 0);

if(buffer\_parent == (void\*)-1){

printf("Failed to attach shared memory of producer\n");

return -1;

}

//execute producer function

producer(buffer\_parent);

//wait until child buffer is detached

wait(NULL);

//detach parent buffer form shared memor

if(shmdt((void\*)buffer\_parent) == -1)

printf("Failed to detach shared memory of producer\n");

//deallocate shared memory

if(shmctl(shm\_id, IPC\_RMID, 0) == -1){

printf("Failed to deallocate shared memory\n");

return -1;

}

else{

printf("Success to deallocate shared memory\n");

return 0;

}

}

}

void producer(Buffer \* buffer){

char \* input;

int i;

for(i=0; i<NUM\_INPUT; i++){

printf("Input a string: ");

scanf("%s",input);

while(((buffer->in+1)%BUFFER\_SIZE) == buffer->out)

; //do nothing

strncpy(buffer->data[buffer->in], input, MAX\_LEN-1);

buffer->data[buffer->in][strlen(input)] = 0;

buffer->in = (buffer->in+1) % BUFFER\_SIZE;

printf("producer inserting \"%s\"\n", input);

}

}

void consumer(Buffer \* buffer){

char \* input;

int i;

for(i=0; i<NUM\_INPUT; i++){

while(buffer->in == buffer->out)

; //do nothing

strcpy(input, buffer->data[buffer->out]);

buffer->out = (buffer->out+1) % BUFFER\_SIZE;

sleep(2);

printf("\n\t\tConsumer deleted \"%s\"\n", input);

}

}