Implement dynamic circular queue in Linux char device, which takes data from IOCTL calls.

In Kernel Space:

IOCTL operations are:

SET\_SIZE\_OF\_QUEUE: which takes an integer argument and creates queue according to given size

PUSH\_DATA: passing a structure, which contains data and its length, and push the data of given length

POP\_DATA: passing a structure same as above and just pass the length, while popping data in the structure can be random.

In user space:

Demonstrate the use of above char device, with sys IOCTL calls. Make sure to make this device blocking i.e. if there is no data passed while popping it should wait until other process pushes the data into the char device. The device should be /dev/<your\_name>.

Example of the userspace driver:

-configurator.c

#include <stdio.h>

#include <fcntl.h>

#include <unistd.h>

#include <sys/ioctl.h>

#define DRIVER\_NAME "/dev/vicharak"

#define SET\_SIZE\_OF\_QUEUE \_IOW('a', 'a', int \* )

int main(void) {

int fd = open(DRIVER\_NAME, O\_RDWR);

int size = 100;

int ret = ioctl(fd, SET\_SIZE\_OF\_QUEUE, & size);

close(fd);

return ret;

}

- filler.c

#include <stdio.h>

#include <fcntl.h>

#include <unistd.h>

#include <sys/ioctl.h>

#include <errno.h>

#include <string.h>

#define DRIVER\_NAME "/dev/vicharak"

#define PUSH\_DATA \_IOW('a', 'b', struct data \* )

struct data {

int length;

char \* data;

};

int main(void) {

int fd = open(DRIVER\_NAME, O\_RDWR);

//simple if statement to treat empty queue.

if (fd < 0) {

printf("Error opening device: %s\n", strerror(errno));

return 1;

}

struct data \* d = malloc(sizeof(struct data));

d.length = 3;

d.data = malloc(3);

memcpy(d.data, "xyz", 3);

int ret = ioctl(fd, PUSH\_DATA, d);

//if statement for fail error.

if (ret < 0) {

printf("Error popping data: %s\n", strerror(errno));

}

close(fd);

free(d.data);

free(d);

return ret;

}

- reader.c

#include <stdio.h>

#include <fcntl.h>

#include <unistd.h>

#include <sys/ioctl.h>

#include <stdlib.h>

#include <string.h>

#include <errno.h>

#define DRIVER\_NAME "/dev/vicharak"

#define POP\_DATA \_IOR('a', 'c', struct data \* )

struct data {

int length;

char \* data;

};

int main(void) {

int fd = open(DRIVER\_NAME, O\_RDWR);

if statement

if (fd < 0) {

printf("Error opening device: %s\n", strerror(errno));

return 1;

}

struct data \* d = malloc(sizeof(struct data));

if statement

if (!d) {

printf("Memory allocation failed for struct: %s\n", strerror(errno));

return 1;

}

d.length = 3;

d.data = malloc(3);

//if statement

f (!d->data) {

printf("Memory allocation failed for data buffer: %s\n", strerror(errno));

return 1;

}

int ret = ioctl(fd, PUSH\_DATA (POP\_DATA), d);

// if statement

if (ret < 0) {

perror("Failed to pop data");

} else {

printf("Received: %.\*s\n", d->length, d->data);

}

close(fd);

free(d.data);

free(d);

return ret;

}

Kernel driver should accept above IOCTL functions.