1. 개요

두 프로그램 helloxv6과 htac을 각각 구현한다. helloxv6은 표준 출력에 “Hello xv6 World”를 출력하는 프로그램이며, htac은 파일 경로와 정수 n을 입력받으면, 해당 파일의 가장 뒷 줄부터 n개의 줄을 출력하는 프로그램이다. Makefile에 helloxv6과 htac 관련 정보를 적절히 넣어, xv6 빌드 시 프로그램이 함께 컴파일되어 실행할 수 있도록 한다.

2. 설계

두 프로그램 helloxv6와 htac은 각각 별도의 파일에 독립적으로 구현되었다.

2-1. 소스코드별 함수 개요

2-1-1. helloxv6.c

프로그램 helloxv6을 구현한 소스 파일

2-1-1-1. int main(int argc, char **argv)

표준 출력(file descriptor 1)에 Hello xv6 World(개행)을 출력한 후 프로그램을 종료한다.

2-1-2. htac.c

프로그램 htac을 구현한 소스 파일

2-1-2-1. int main(int argc, char **argv)

argv에서는 2개의 인자를 받는다. 인자는 각각 파일의 경로와 정수 n을 의미하며, 전역 변수 line에 정수 n을 할당한 후, 파일의 디스크립터 fd에 대하여 htac(fd) 함수를 호출한다.

2-1-2-2. void htac(int fd)

파일 디스크립터 fd가 주어지면, 해당 파일의 맨 뒤부터 line개의 줄을 읽어 출력하는 함수이다.

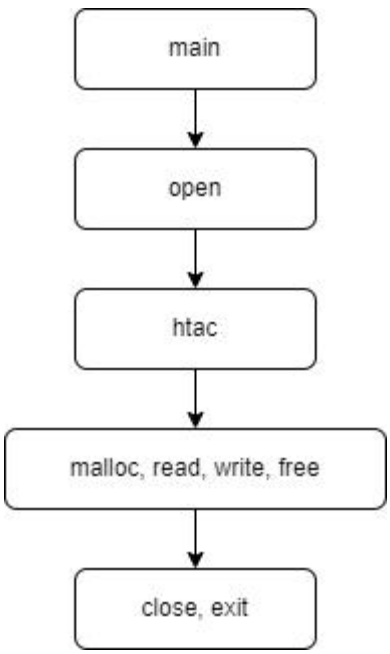
2-2. 함수 호출 그래프

2-2-1. helloxv6.c

main (helloxv6.c) -> write, exit (user.h)

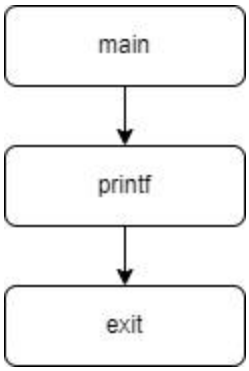
2-2-2. htac.c

main, htac은 htac.c에 정의된 함수이고, 그 외 함수는 user.h에 정의된 함수이다.

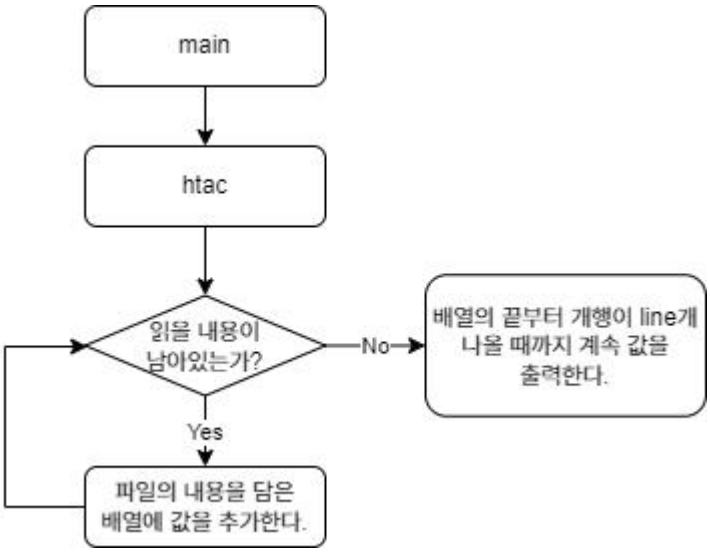


2-3. 순서도

2-3-1. helloxv6 순서도



2-3-2. htac 순서도



3. 실행결과

3-1. helloxv6

```
sb: size 1000 nblock
init: starting sh
$ helloxv6
Hello xv6 World
$
```

3-2. htac

3-2-1. htac 기본 실행

```
$ htac 5 README
simulator and run "make qemu".
Then run "make TOOLPREFIX=i386-jos-elf-". Now install the QEMU PC
x86 ELF binaries (see https://pdos.csail.mit.edu/6.828/).
will need to install a cross-compiler gcc suite capable of producing
"make". On non-x86 or non-ELF machines (like OS X, even on x86), you
$
```

3-2-2. htac 파일에 빈 줄이 있는 경우

```
$ htac 10 README
simulator and run "make qemu".
Then run "make TOOLPREFIX=i386-jos-elf-". Now install the QEMU PC
x86 ELF binaries (see https://pdos.csail.mit.edu/6.828/).
will need to install a cross-compiler gcc suite capable of producing
"make". On non-x86 or non-ELF machines (like OS X, even on x86), you
To build xv6 on an x86 ELF machine (like Linux or FreeBSD), run

BUILDING AND RUNNING XV6

We don't process error reports (see note on top of this file).
$
```

3-2-3. htac 파일이 존재하지 않는 경우

```
$ htac 5 Aasdf
htac: cannot open Aasdf
$
```

3-2-4. line이 1 미만인 정수인 경우

```
$ htac -1 README
$ htac 0 README
$
```

3-2-5. line 수가 파일 라인보다 많은 경우

```
$ htac 9999999999 README
simulator and run "make qemu".
Then run "make TOOLPREFIX=i386-jos-elf-". Now install the QEMU PC
x86 ELF binaries (see https://pdos.csail.mit.edu/6.828/).
will need to install a cross-compiler gcc suite capable of producing
```

(중간 생략)

```
but is implemented for a modern x86-based multiprocessor using ANSI C.
Version 6 (v6).  xv6 loosely follows the structure and style of v6,
xv6 is a re-implementation of Dennis Ritchie's and Ken Thompson's Unix

(https://github.com/mit-pdos/xv6-riscv.git)
our efforts to the RISC-V version
NOTE: we have stopped maintaining the x86 version of xv6, and switched
$
```

3-2-6. htac 인자가 부족한 경우

```
$ htac 2
$ htac 3 README
simulator and run "make qemu".
Then run "make TOOLPREFIX=i386-jos-elf-". Now install the QEMU PC
x86 ELF binaries (see https://pdos.csail.mit.edu/6.828/).
$
```

4. 소스코드 (주석 포함)

4-1. helloxv6 (helloxv6.c)

```
#include "types.h"
#include "user.h"
int main(int argc, char **argv) {
    printf(1, "Hello xv6 World\n"); // stdout
    exit(); // exit program.
}
```

4-2. htac (htac.c)

```
#include "types.h"
#include "user.h"
int line; // count that prints lines from back
char buf[512]; // temp buffer
void htac(int fd) {
    int line_count =0, data_sz =0;
    char *data =0, *nxt;
    {
        int n;
        while ((n = read(fd, buf, 512)) >0) { // read until EOF
            for (int i =0; i < n; i++) line_count += buf[i] =='\n'; // counts '\n'
            nxt =malloc(data_sz + n); // there is no realloc, so allocate new space, and move data to new space from previous space.
            for (int i =0; i < data_sz; i++) nxt[i] = data[i];
            if (data) free(data); // free previous space.
            for (int i =0; i < n; i++) nxt[i + data_sz] = buf[i]; // push back data from buffer
            data = nxt;
            data_sz += n;
        }
    }
    for (int i = line_count; i >=0 && i > line_count - line; i--) { // read from back
        int curr =0;
        for (int j =0; j < data_sz; j++) {
            if (data[j] =='\n') curr++; // if meet '\n', add increase current line count.
            else if (i == curr) write(1, data + j, 1); // write current character to stdout when current line equals "i".
        }
        write(1, "\n", 1); // print new line.
    }
    free(data);
}
int main(int argc, char **argv) {
    int fd, i;
    if (argc <=1) {
        htac(0);
        exit();
    }
```

```
    for(i =1; i +1 < argc; i +=2){
        line = atoi(argv[i]); // char* to int.
        if((fd = open(argv[i +1], 0)) <0){ // open file
            printf(1, "htac: cannot open %s\n", argv[i +1]);
            exit();
        }
        htac(fd); // call htac function with file descriptor
        close(fd); // close file
    }
    exit(); // exit program.
}
```

4-3. Makefile (수정한 부분만)

4-3-1. UPROGS

```
UPROGS=\
        _cat\
        _echo\
        _forktest\
        _grep\
        _init\
        _kill\
        _ln\
        _ls\
        _mkdir\
        _rm\
        _sh\
        _stressfs\
        _usertests\
        _wc\
        _zombie\
        _helloxv6\
        _htac
```