Document coping with System Test for NPS System

- **Test Cases Specification**
- **Test Summary Report**

Project Team
Team 3

Latest update on:
2015-12-08

Team Information
201411256 강유민
201411258 강태준
201411265 김서우
201411315 정유환
201411321 홍유리

Table of Contents

1 Introduction

1.1 Objectives

본 문서는 Network Printer System의 System Test를 수행 후 결과를 바탕으로 대응한 방법에 대한 문서이다.

1.2 References

NPS STR (by Team2)

NPS STR (by Team3)

2 Fail List

2.1 Fail Result of system testing by team 2

Test Case Number	Description
T2_01	대기 열에 userID 미 표시
T2_02	사용자 등록 시 admin 아이디가 붙어서 저장
T2_03	사용자 등록 시 길이가 긴 문자열 입력 시 프로그램 종료
T2_04	프린트 중단 시 현재까지 출력된 파일이 출력되지 않음
T2_05	1분 안에 여러 번 출력 요청 시 파일이 덮어씌워 생성됨
T2_06	파일 한 줄에 40자 입력 후 출력 요청 시 프로그램 종료 파일이 11줄 일 때 2장으로 인식하는가 - 14줄까지 한 장으로 인식 290글자 출력 시 잉크가 두 번 소모

2.2 Fail Result of system testing by team 3

Test Case Number	Description
T3_01	충전이 끝나고 대기하던 인쇄가 진행 될 때 대기자 수가 줄어들지 않는다.
T3_02	dslab이라고 userlist.txt에 쓰이긴 하지만 다음 줄에 쓰이지 않아서 기존의 마지막 ID 뒤에 이어서 써진다.
T3_03	이미 충전 중이라는 알람이 뜨지 않는다.
T3_04	최대치로 충전한다는 알람은 뜨나, 최대치로 충전되지는 않는다.

3 Actions dealing with fails

3.1 Actions dealing with fails by Team 2

3.1.1 T2_01

Team3는 개발할 때에 LCD 화면에 대기자의 ID를 표시하는 것이 아닌, 현재 대기자 수를 출력하는 방향으로 개발하였다. 따라서 User Manual에 위의 내용을 추가하였다.

3.1.2 T2_02

사용자 등록 부분의 기존의 코드

```
void user_join(char * ID)
{
    FILE *fp = fopen("../userlist.txt", "a");

    fprintf(fp, "%s", ID);

    fclose(fp);
}
```

사용자 등록 부분의 수정된 코드

```
void user_join(char * ID)
{
    FILE *fp = fopen("../userlist.txt", "a");

    fprintf(fp, "%s\n", ID);

    fclose(fp);
}
```

개행 문자('₩n')를 추가하여 기존의 userlist.txt에 있던 마지막 ID 바로 옆에 새로운 ID가 붙지 않도록 수정하였다.

3.1.3 T2_03

Team3는 개발할 때에 ID의 길이에 제한을 두었다. ID는 최대 10글자가 가능하다. 따라서 User Manual에 위의 내용을 추가하였다.

3.1.4 T2_04

인쇄물 출력 부분의 기존의 코드

```
while(fgets(one_line, 499, original_file) != NULL)
{
    PS.print_current_state = 1; // 인쇄 중으로 상태 변경
    if(read_update()) // 새로운 값 입력 들어옴
    {
        int now_command;
        now_command = read_commmand();

        switch(now_command)
        {
            case 1 :
            printcontroller(now_command);
            break;
            case 2 :
            return;
            case 3 :
            case 4 :
            refillcontroller(now_command);
            break;
            case 5 :
            case 6 :
            case 7 :
            manageusercontroller(now_command);
            break;
```

인쇄물 출력 부분의 수정된 코드

```
while(fgets(one_line, 499, original_file) != NULL)
{
    PS.print_current_state = 1; // 인쇄 중으로 상태 변경
    if(read_update()) // 새로운 값 입력 들어옴
    {
        int now_command;
        now_command = read_commmand();

        switch(now_command)
        {
            case 1 :
            printcontroller(now_command);
            break;
            case 2 :
            fclose(new_file)
            return;
            case 3 :
            case 4 :
            refillcontroller(now_command);
            break;
            case 5 :
            case 6 :
            case 7 :
            manageusercontroller(now_command);
            break;
        }
```

파일 출력 도중 Stop 명령이 들어왔을 때 인쇄 중이던 파일을 fopen으로 열려 있었는데 fclose를 하지 않아서 출력 도중에 인쇄를 멈추었을 때 진행 상황이 출력되지 않는 것을 확인하여 fclose를 추가하였다.

3.1.5 T2_05

출력 파일 생성 부분의 기존의 코드

```
char buf_year[5];
char buf_mon[3];
char buf_mday[3];
char buf_hour[3];
char buf_min[3];
char new_file_name[13];

original_file = fopen(WT.file, "r");

timer = time(NULL);

t = localtime(&timer);

n=sprintf (buf_year, "%d",t->tm_year+1900);
n=sprintf (buf_mon, "%d",t->tm_mon+1);
n=sprintf (buf_mday, "%d",t->tm_mday);
n=sprintf (buf_hour, "%d",t->tm_hour);
n=sprintf (buf_min, "%d",t->tm_min);


sprintf(new_file_name,"../%d%d%d%d%d.txt", t->tm_year+1900, t->tm_mon+1, t->tm_mday, t->tm_hour, t->tm_min);
```

출력 파일 생성 부분의 수정된 코드

```
char buf_year[5];
char buf_mon[3];
char buf_mday[3];
char buf_hour[3];
char buf_min[3];
char buf_sec[3];
char new_file_name[13];

original_file = fopen(WT.file, "r");

timer = time(NULL);

t = localtime(&timer);

n=sprintf (buf_year, "%d",t->tm_year+1900);
n=sprintf (buf_mon, "%d",t->tm_mon+1);
n=sprintf (buf_mday, "%d",t->tm_mday);
n=sprintf (buf_hour, "%d",t->tm_hour);
n=sprintf (buf_min, "%d",t->tm_min);
n=sprintf (buf_sec, "%d", t->tm_sec);

sprintf(new_file_name,"../%d%d%d%d%d%d.txt", t->tm_year+1900, t->tm_mon+1, t->tm_mday, t->tm_hour, t->tm_min, t->tm_sec);
```

SRS를 보고 출력 파일의 이름이 년월일시분.txt 인줄 알았는데 그런 이름 생성 방식을 쓰니 빠르게 출력되는 파일의 이름이 겹쳐진다는 점을 알았다. 따라서 년월일시분초.txt로 생성되는 파일의 이름을 바꾸었다.

3.1.6 T2_06

출력물 데이터 계산 부분의 기존의 코드

```
while (fgets(line, 255, file) != NULL)
{

        // 한 줄을 읽어 저장(마지막에 자동으로 종료코드 삽입)
        Num_of_Line++; // 줄 수 증가
        for (i = 0; i < 30; i++) // 한 줄의 길이 만큼 반복문
        {
                ch = line[i];
                if ((ch>32) && (ch<127)) // 알파벳이나 특수문자 인가?
                        Num_of_Alpha++;
                else if (ch == 32) // 공백인가?
                        Num_of_Space++;


        }
}
```

출력물 데이터 계산 부분의 수정된 코드

```
while (fgets(line, 255, file) != NULL)
{

        // 한 줄을 읽어 저장(마지막에 자동으로 종료코드 삽입)
        Num_of_Line++; // 줄 수 증가
        for (i = 0; i < 30; i++) // 한 줄의 길이 만큼 반복문
        {
                ch = line[i];
                if ((ch>32) && (ch<127)) // 알파벳이나 특수문자 인가?
                        Num_of_Alpha++;
                else if (ch == 32) // 공백인가?
                        Num_of_Space++;

                if(line[i] == '\n')
                {
                        break;
                }
        }
}
```

기존의 방식에서는 개행 문자를 만나도 while문을 멈추지 않아서 필요한 종이량 계산이 잘못되고, 필요한 잉크량 계산이 잘못 되었었다. 이를 막기 위해서 개행문자를 만나면 반복문을 탈출하는 부분을 추가하였다.

3.2 Actions dealing with fails by Team 3

3.2.1 T3_01

인쇄 목록 읽기 부분의 기존의 코드

```
while(fgets(read_wait_line, 49, fp) != NULL)
{
    temp_file_pointer = ftell(fp);
    fclose(fp);
    char * temp_paper = (char *)malloc(sizeof(char)*10);
    WT.ID = strtok(read_wait_line,",");
    WT.file = strtok(NULL,",");
    temp_paper = strtok(NULL,",");
    WT.paper = atoi(temp_paper);
     PS.print_current_state = 1;

    printinterface(WT);

    if(PS.wait_num > 0)
    {
        PS.wait_num --;
    }
```

인쇄 목록 읽기 부분의 수정된 코드

```
while(fgets(read_wait_line, 49, fp) != NULL)
{
    temp_file_pointer = ftell(fp);
    fclose(fp);
    char * temp_paper = (char *)malloc(sizeof(char)*10);
    WT.ID = strtok(read_wait_line,",");
    WT.file = strtok(NULL,",");
    temp_paper = strtok(NULL,",");
    WT.paper = atoi(temp_paper);
     PS.print_current_state = 1;

    if(PS.wait_num > 0)
    {
        PS.wait_num --;
    }

    printinterface(WT);
```

실제로 인쇄를 진행하는 printinterface에 진입하고 난 후에 대기자의 수를 줄이는 점을 고쳐서 printinterface로 진행하기 전 대기자 수를 줄이는 방향으로 수정하였다.

3.2.2 T3_02

3.1.2 T2_02 부분과 동일

3.2.3 T3_03

에러나 알람이 발생할 때 LCD에 표시해 주는 기존의 코드

```
        break;
        case 6:
        // 충전하면 최대 잉크량 초과할 때
        printf(" Warning !! \n");
        printf(" 잉크를 충전하면 최대 잉크 량을 초과합니다. \n");
        printf(" 잉크를 최대치로 충전합니다. \n");
        break;
        case 7:
        // 유저 추가, 삭제, 보여주기 명령 했는데 휴식 중이 아닐 때
        printf(" Warning !! \n");
        printf(" 휴식 중이 아니므로 명령을 수행할 수 없습니다. \n");
        break;
        case 8:
        // 등록할 사람이 이미 존재 할 때
        printf(" Warning !! \n");
        printf(" 등록할 사용자가 이미 존재합니다. \n");
        break;
        case 9:
        // 삭제할 사람이 존재 하지 않을 때
        printf(" Warning !! \n");
        printf(" 삭제할 사용자가 존재하지 않습니다. \n");
        break;

    }
```

에러나 알람이 발생하였을 때 LCD에 표시해 주는 수정된 코드

```
    case 7:
    // 유저 추가, 삭제, 보여주기 명령 했는데 휴식 중이 아닐 때
    printf(" Warning !! \n");
    printf(" 휴식 중이 아니므로 명령을 수행할 수 없습니다. \n");
    break;
    case 8:
    // 등록할 사람이 이미 존재 할 때
    printf(" Warning !! \n");
    printf(" 등록할 사용자가 이미 존재합니다. \n");
    break;
    case 9:
    // 삭제할 사람이 존재 하지 않을 때
    printf(" Warning !! \n");
    printf(" 삭제할 사용자가 존재하지 않습니다. \n");
    break;
    case 10:
    // 충전 요청인데 이미 충전 중
    printf(" Warning !! \n");
    printf(" 현재 충전 중이므로 또 다른 충전을 수행할 수 없습니다. \n");
}
```

현재 충전 중인데 사용자가 또 충전 명령을 보냈을 때를 고려하지 않은 점을 깨달아 새로운 알람을 추가하였다.

3.2.4 T3_04

잉크, 용지 충전 부분의 기존의 코드

```
void refill_ink(int amount)
{
    int i;
    PS.print_current_state = 2;

    for(i = 1; i<(amount+1); i++) // amount 만큼 반복
    {
         if(read_update()) // 새로운 값 입력 들어옴
        {
            int now_command;
            now_command = read_commmand();

            switch(now_command)
            {
                case 1 :
                case 2 :
                printcontroller(now_command);
                break;
                case 3 :
                case 4 :
                refillcontroller(now_command);
                break;
                case 5 :
                case 6 :
                case 7 :
                manageusercontroller(now_command);
                break;
            }
        }
        PS.print_current_state = 2;
        PS.temp_ink;
        PS.current_ink ++;
        if(((i%100) == 0) || (amount == i)) // 100 충전했거나, 마지막 충
        {
            lcdinterface("admin", 0, 0); // lcd 내부에서 sleep
        }
    }
    if(PS.wait_num >0)
    {
        read_wait();
    }
}
```

```
void refill_paper(int amount)
{
    int i;
    PS.print_current_state = 2;

    for(i = 1; i<(amount+1); i++) // amount 만큼 반복
    {
        if(read_update()) // 새로운 값 입력 들어옴
        {
            int now_command;
            now_command = read_commmand();

            switch(now_command)
            {
                case 1 :
                case 2 :
                printcontroller(now_command);
                break;
                case 3 :
                case 4 :
                refillcontroller(now_command);
                break;
                case 5 :
                case 6 :
                case 7 :
                manageusercontroller(now_command);
                break;
            }
        }
        PS.print_current_state = 2;
        PS.temp_paper;
        PS.current_paper ++;
        if(((i%10) == 0) || (amount == i)) // 10장 충전했거나, 마지막
        {
            lcdinterface("admin", 0, 0); // lcd 내부에서 sleep
        }
    }
    if(PS.wait_num >0)
    {
        read_wait();
    }
```

잉크, 용지 충전 부분의 수정된 코드

```
void refill_ink(int amount)
{
    int i;
    PS.print_current_state = 2;
    if((amount + PS.current_ink) > MAX_INK) // 최대치로 충전
    {
        amount = MAX_INK - (PS.current_ink);
    }

    for(i = 1; i<(amount+1); i++) // amount 만큼 반복
    {
         if(read_update()) // 새로운 값 입력 들어옴
        {
            int now_command;
            now_command = read_commmand();

            switch(now_command)
            {
                case 1 :
                case 2 :
                printcontroller(now_command);
                break;
                case 3 :
                case 4 :
                refillcontroller(now_command);
                break;
                case 5 :
                case 6 :
                case 7 :
                manageusercontroller(now_command);
                break;
            }
        }
        PS.print_current_state = 2;
        PS.temp_ink ++;
        PS.current_ink ++;
        if(((i%100) == 0) || (amount == i)) // 100 충전했거나, 마지막
        {
            lcdinterface("admin", 0, 0); // lcd 내부에서 sleep
        }
    }
    if(PS.wait_num >0)
    {
        read_wait();
    }
```

```
{
    int i;
    PS.print_current_state = 2;
    if((amount + PS.current_paper) > MAX_PAPER)
    {
        amount = MAX_PAPER - (PS.current_paper);
    }

    for(i = 1; i<(amount+1); i++) // amount 만큼 반복
    {
        if(read_update()) // 새로운 값 입력 들어옴
        {
            int now_command;
            now_command = read_commmand();

            switch(now_command)
            {
                case 1 :
                case 2 :
                printcontroller(now_command);
                break;
                case 3 :
                case 4 :
                refillcontroller(now_command);
                break;
                case 5 :
                case 6 :
                case 7 :
                manageusercontroller(now_command);
                break;
            }
        }
        PS.print_current_state = 2;
        PS.temp_paper++;
        PS.current_paper ++;
        if(((i%10) == 0) || (amount == i)) // 10장 충전했거나, 마지막 충전 일때
        {
            lcdinterface("admin", 0, 0); // lcd 내부에서 sleep
        }
    }
    if(PS.wait_num >0)
    {
        read_wait();
    }
}
```

충전 값으로 최대 치를 넘는 값이 들어오는 상황을 생각 안하고 개발을 하여서 if 문을 추가하여 최대 치를 넘는 다면 최대치에서 현재의 양을 뺀 만큼 충전을 하도록 하였다. 또한 임시로 현재 잉크와 용지 양을 저장해 놓는 temp_paper와 temp_ink를 충전 시에 증가시키지 않아서 인쇄 조건이 제대로 계산이 안될 것 같아서 수정하였다.