**Documentation About Methods Used In [Plan\_it] Project**

Authors: 2023011393 Nawon Kim, 2021114026 Jeongwoo Kim, 2023013565 Dahye Jeong

Version 0.0.2 이후의 버전부터 추가되거나 수정된 사항만 반영되어 있습니다.

**0.0.3 수정사항**

(Core)

삽입시 기존에 북마크가 있을 경우 새 자료 무시

윤년, 28, 29, 30, 31일 구분 및 잘못된 날짜를 보유한 데이터는 처음부터 생성되지 않도록 조치

human readable 파일 생성시, 3년의 인터벌을 두는 범위(2023-2025 등)의 경우 첫 연도의 트리가 잘못 형성되는 오류를 제거

해시값 구성 조건 새로 구성

(UX)

좌표 공간 새로 구성

하이라이트 기능 수정

**0.0.5 수정사항**

(Core) 레코드 삭제 기능 구현

(Core) 반복 세이브/로드시 메모리 누수 문제 해결, 레코드 필드에 대한 동적 할당 해제 기능 적용

(Core) getTodaySchedule\_계열 함수 반환형 int로 업데이트, 반환 데이터가 의미를 가짐.

(Core) sighandler 및 itimer 기반 리마인더 기능 구현

(Core) 리마인더 세이브/로드 기능 구현(저장 파일과 함께 저장됨), Human-readable 파일 내에도 Reminder 정보 제공

(Core) 현재 실행 중인 리마인더 삭제 / 확인 기능 구현

(Core) 실행시 Core부 초기화 함수 coreInit() 내부적 업데이트

(Core) 일정 수정 기능 소급 적용

(UX) Core부 함수 참조 부분에 대해 반환형 개선 및 신규 함수 추가

**0.0.6 수정사항**

(Core) UX 화면 뿐만 아니라 -d 옵션으로 진입 가능한 디버깅 메뉴에서도 띄어쓰기가 포함된 문자열 입력 가능

(Core) 수정 기능 관련 함수 구현 중

(Core) 양방향 iterator 기반 상세 일정 순회 기능으로 확장

(UX) 일정 세부사항 조회 기능 추가

(UX) 일정 세부사항 조회에 양방향 iterator 기반 양방향 순회 추가

(UX) 문자열 길이 측정 관련 GDB 기반 오류 현재 파악, 해결 위해 노력 중 (아래 별첨 참고)

(기타) 랜덤 레코드 생성 프로그램 내부 랜덤 문자열 업데이트, 세이브파일 업데이트

**0.0.7 수정사항**

(Core) 띄어쓰기가 포함된 문자열 기능 안정화

(Core) 동적 할당 해제 알고리즘 개선

(Core) UX레이어와의 상호작용을 위해 일부 함수 내부 구조 개선

(UX) 메뉴 심도 3단계 -> 2단계로 단순화

(UX) 양방향 순회 방식 일자 탐색, 선택되어 있는 대상을 바로바로 지울 수 있도록 개선 (메뉴 심도의 단축과도 관련 있는 기능)

(UX) 삽입 함수를 낮취진 심도인 2단계에 삽입

(UX) 일정 개수 출력 함수의 string refresh 관련 알고리즘 개선

(UX) details string parser 알고리즘 개선

**Week 2 수정사항**

**UX 파트 (plnit\_uxCore.c)**

void clearGivenCalendarArea(int row, int col);

* 6 by 7 구조의 달력의 각 칸을 지정하여, 그 칸의 모든 내용을 지웁니다.

void clearGivenRowCols(int fromRow, int fromCol, int toRow, int toCol);

* clearGivenNonCalendarArea에서 내부적으로 사용되는 함수입니다.

void clearGivenNonCalendarArea(/\*pre-defined Macros\*/int area);

* SLL, SLC 등의 영역을 매크로(#define)로 지정하였기에,

clearGivenNonCalendarArea(SLC) 등의 방식으로 사용 가능합니다.

void save\_UXPart(void);

* 저장되었음을 알리는 UX적인 기능을 추가적으로 발휘하며, 저장 액션을 수행하는 함수입니다.

Critical part인 만큼, sigprocmask등으로 시그널 블로킹이 필요 해 보입니다.

void load\_UXPart(void);

* 로드되었음을 알리는 UX적인 기능을 추가적으로 발휘하며, 로드 액션을 수행하는 함수입니다.

Critical part인 만큼, sigprocmask등으로 시그널 블로킹이 필요 해 보입니다.

void setInputModeSigHandler(int status);

* 입력 모드 수행 중 나가기 동작을 수행하기 위한 signal handling method입니다.

void inputMode\_sigHndl(int signum);

* 입력 모드용 signal handler 입니다.

void popup(char\* title, char\* str1, char\* str2, int delay)

* title에는 알림의 제목을, str1과 str2는 알림의 내용을 표시 할 수 있습니다. Delay는 초 단위로 입력받으며, 현재 프로세스에 미리 알림 등으로 인해 itimer가 설정 되었을 경우, 충돌을 피하기 위해 fork()로 구현되어 있습니다.

void edit\_plan(unsigned long long targetDate, int\* page);

* ??? int\* page?

**Core 파트 (plnit\_dbCore.c)**

void coreInit(void)

* UX 엔진 부팅시 DB 엔진 초기화 함수입니다.

void getBookMarkedInDate(unsigned long long today, int counter, char\* str)

* UX부를 위한 북마크 스트링 추상화 함수입니다. toDoPtr getBookMarked(unsigned long long src, int distance) 및 해시함수를 기반으로 작동합니다. counter에는 오늘을 기준으로 앞으로 몇 개의 북마크를 보여줄 것인지에 해당하는 데이터가 들어갑니다.
* \*[^15:30]^Sample\_Title: 북마크 된, 시간이 15:30인, 제목이 Sample\_Title인 스트링
* 북마크 된 것만 불러오는데, 북마크 prefix code가 들어간 이유는 아래 format string 코드 재활용을 하여 작성이 수월하도록, 도와드리기 위함입니다.

void deleteWhileIterate(unsigned long long src, int pageNum);

* YYYYMMDD식의 src의 iterable page number을 기준으로(주의: index 아님) 삭제합니다.

int editWhileIterate(unsigned long long src, int pageNum);

* YYYYMMDD식의 src의 iterable page number을 기준으로(주의: index 아님) 수정합니다. 다만 아직 미구현 상태입니다.

int setReminder(time\_t current, time\_t delta, int repeatCnter, char\* what, int intervals);

* Current는 time(NULL)을 넣으며(저장 기능의 용이성을 위해 있는 인자라서, 그냥 활용하실 때는 time(NULL)을 넣으시면 됩니다.), delta는 현재 기준으로 n초 후의 미리 알림을 설정하겠다는 뜻이며, repeatCnter과 intervals는 미리 알림 종료 시점 기준으로 얼마나 전부터 몇번 미리 알림을 띄우겠다 라는 의미입니다.
* 즉, 지금이 1시고 2시에 있는 약속을 30분 전부터 3번 미리 알림을 띄우고자 하면, 1시 30분, 1시 40분, 1시 50분에 미리 알림이 뜰 것입니다.
* What은 내용 string이며, 30자 제한입니다.

int isReminderSetAlready(char\* str);

* 리마인더가 설정되어 있다면, 1을 반환함과 동시에 str에 앞서 말한 What이 저장됩니다.
* 라마인더가 설정되어 있지 않으면, 0을 반환합니다.

void turnOffReminder(void);

* 호출시 리마인더를 끕니다.

int getTodaySchedule\_Summarized(unsigned long long today, char\* strbuf);

* 업데이트 되었습니다. 삭제 기능이 구현된 만큼, 해당 자리에 레코드가 존재하지 않는다면 -1을 리턴하고 buf에 “no\_data”를 기록합니다.
* 레코드가 있다면, 0을 반환합니다.

int getTodaySchedule\_withDetails(unsigned long long today, char\* strbuf);

* 업데이트 되었습니다. 삭제 기능이 구현된 만큼, 해당 자리에 레코드가 존재하지 않는다면 -1을 리턴하고 buf에 “no\_data”를 기록합니다.
* 레코드가 있다면, 0을 반환합니다.

int setSchedule(unsigned long long today, char\* title, char\* details, int priority)

* 정상적으로 삽입이 완료되면 0
* 북마크가 겹치면 1
* 유효하지 않은 날짜는 2를 반환합니다.

void reminderHandler(int signum)

* 기능 구현 여부를 보이기 위해 CLI상에서 구동되지만, 실제로는 UX에서 이루어져야 하는 시그널 핸들링입니다. 구현 방식과 가이드라인은 dbCore 파일 내 각주를 참고해 주시면 감사하겠습니다.

Format string related

* 조언 해 주신대로 prefix code를 단순화 하여서, time, title, detail, bookmark 여부를 다음과 같은 논리에 기반하여 format string을 리턴합니다. (new line 여부는 ux단에서 해결합니다)
* [^15:30\*]^Sample\_Title: 북마크\* 된, 시간이 15:30인, 제목이 Sample\_Title인 스트링
* [^15:30]^Sample\_Title: 시간이 15:30인, 제목이 Sample\_Title인 스트링
* [^15:30\*]^%Sample\_Title]]Sample\_Details: 북마크\* 된, 시간이 15:30이고, 제목이 Sample\_Title이며, 내용이 Sample\_Details
* [^15:30]^%Sample\_Title]]Sample\_Details: 시간이 15:30이고, 제목이 Sample\_Title이며, 내용이 Sample\_Details

int editWhileIterate(unsigned long long src, int pageNum, unsigned long long t\_day, char\* t\_title, char\* t\_details, int t\_priority) => 폐기

* 내부적으로 삭제 후 편집 대상 삽입의 방식으로 구현되어 있습니다. 단, 과정이 아래와 같습니다.
* 편집 대상 페이지가 존재하지 않는 경우: 2 리턴
* 편집을 시도하려고 하였으나, 편집 대상이 새로 북마크된 속성을 가지도록 하였는데 이미 그 일자에 북마크된 일정이 존재할 때: 1 리턴
* 정상적으로 “삭제 후 삽입”이 일어났을 때: 0 리턴
* 기본적으로 deleteWhileIterate에 정보 추가 기능이 있다고 생각하시면 되고, 그러한 사고방식으로 함수 인자에 접근하시면 됩니다.
* setSchedule, editWhileIterate 등 리턴값으로 오류 정보를 알아낼 수 있는 경우 적절한 텍스트로 핸들링 해 주시면 감사하겠습니다.

int getTodaySchedule\_withDetails(unsigned long long today, char\* strbuf, int direction)

* 순회 방향 지정 가능하도록 direction 인자 추가

int editWhileIterate(unsigned long long src, int pageNum, unsigned long long t\_day, char\* t\_title, char\* t\_details, int t\_priority)

* Src와 pagenum은 \_\_whileIterate() 시리즈와 동일하나, 자료를 새로 만들때와 동일한 수준의 인자인 day, title, details. Bookmark no.를 필요로 합니다. 삭제에 기반한 수정이 아니라 실제로 메모리를 수정하도록 기능합니다. 단, 유저의 입력에 따라 수정을 원치 않는 부분도 있을 텐데 그것은 아래와 같이 구분합니다.
* t\_day를 1234로 주면 YYYYMMDDHHMM 수정은 없음
* (중요)title[0]이 NULL이면(즉 null 포인터가 아닌 문자열 배열인데 첫 칸이 널인) title 수정 없음
* (중요)details[0]이 NULL이면 details 수정 없음
* Bookmark(t\_priority)가 -1이면 북마크 여부 수정은 없음
* 리턴값은 정상 수행시 0, 지울게 없으면 2, 북마크 충돌시(즉 기존에 북마크가 존재하는데 수정을 통해 하나 더 생성하려고 할 시) 1입니다.

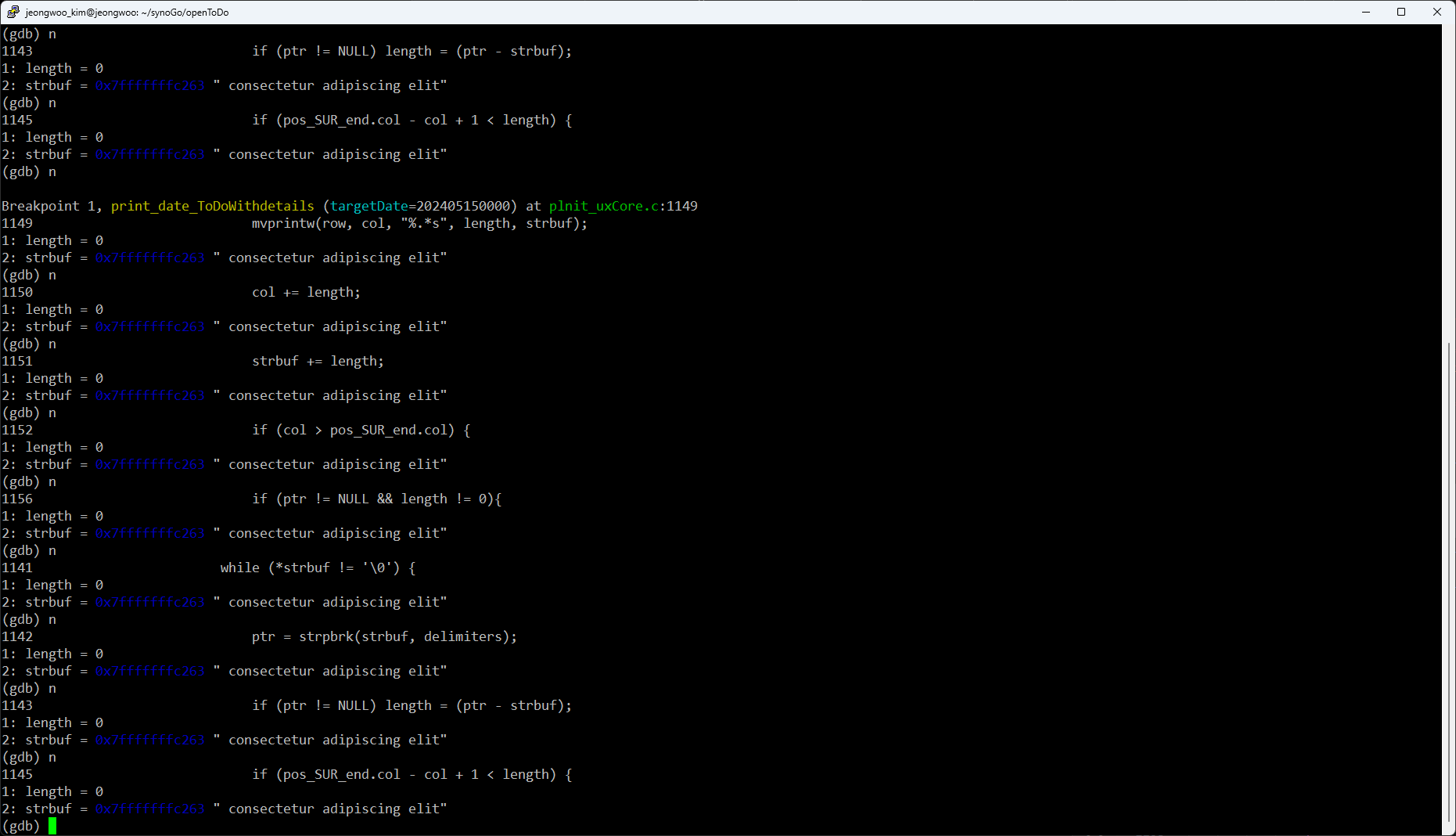
Week 3 회의 이후 새로운 내용

* 북마크가 0, 1, 2, 3 세 가지 종류를 가질 수 있으며, 0은 북마크가 아닌 것이고, 1, 2, 3은 색을 지정합니다. 이때, 북마크가 1일 1개 제한이 있던 것은 동일하게 유지합니다.

**Week 3 수정사항**

1. int isBookMarked(unsigned long long targetDate);
   * YYYYMMDD의 일자를 입력받아, 0(없음), 1, 2, 3은 지정된 색상에 따른 리턴 값을 제공합니다.
2. int setDdayWhileIterate(unsigned long long src, int pageNum);
   * 토의 된 대로, 페이지 열람 화면에서 현재 선택된 페이지에 있는 내용을 d-day의 기준 값으로 삼습니다. (memcpy 기반으로, 원본 레코드가 삭제되어도 남아있습니다.) 따라서 별도의 제목 입력, 날짜 입력이 필요 없어 직관적입니다.
   * 대상 데이터가 존재하지 않으면 2, 이미 2개의 d-day가 설정되어 있으면 1, 정상적으로 작동되었으면 0을 반환합니다. (알림창 요소로도 제공 가능)
   * D+, D- 구분을 별도로 받지 않습니다. 그냥 최대 2개를 저장 할 수 있는 구조입니다. (내부적으로 스택으로 구현되어 있어, 이론상 무한대의 개수를 저장할 수 있긴 하지만 입력단에서 2개로 제한을 걸었습니다.) getDday에서 더 자세하게 다루어집니다.
3. void getDday(int\* slot1, char\* title1, int\* slot2, char\* title2);
   * slot1, 2에는 정수형 변수의 주소를 넘겨주시면 됩니다. 그러면 D -+ 수를 채워서 드립니다.
   * Title 1, 2에는 title 최대 길이 만큼의 버퍼를 넘겨주시면 됩니다.
   * 데이터가 존재하지 않으면 title[0]에 NULL을 채워서 줍니다. 이를 통해 출력 유무를 판단하시면 됩니다.(slot은 그 어떠한 정수도 올 수 있으므로 slot으로 유무 판단은 어렵습니다)
   * 날짜의 차이를 윤년, 각 달의 마지막 일 수 까지 고려하여 정확한 일자의 차이를 구하는 알고리즘이 적용되어 있어, D+, D- 모두에 대응 할 수 있습니다. D+ 2개, D- 2개, D+ 및 D- 각각 하나씩 모두 경우의 수가 될 수 있습니다.
4. void popDday(void);
   * 저장된 d-day를 한 개씩 삭제합니다. 스택 구조라, LIFO임에 주의하셔서 프롬프트 메시지를 띄워주시면 좋을 것 같습니다.
5. int isHoliday(unsigned long long target);
   * YYYYMMDD의 일자를 입력받아, 공휴일이면 1, 일반적인 날(주말 및 평일 포함)이면 0을 반환합니다. 여담으로, 금주 회의에서 토의되었던 내용과는 달리 9의 북마크 값을 지니지 않고, 자료구조상의 개선을 통해(일자 구조체 포인터 관련 개선) 공휴일 여부를 저장합니다. 따라서, 공휴일 일정의 북마크 값은 9가 아닌 0이며, 시간은 토의된 대로 9999로 설정되어 있습니다.
   * 정렬 구조 개선으로, 9999 시간의(공휴일 일정)은 맨 첫 페이지에 오도록 되어 있습니다.
   * 9의 북마크를 지니지 않는 이유는, 내부 로직상 북마크 충돌 여부를 파악할 때 != 0 기반이라, 9로 설정할 시 공휴일에는 북마크 처리가 불가능해지기 때문입니다.
6. randomRecordGenerator의 샘플 개수를 20개로 늘리고, 타이틀 및 텍스트를 마치 실제 존재하는 데이터처럼 만들었습니다.
7. defaultRecordGenerator는 공휴일과 같은 시스템 default 레코드를 읽기 전용으로 생성하는 프로그램이며, 이 프로그램이 생성하는 파일이 ‘public.dsv’ 입니다. 로드시 디폴트로 읽게 되어 있으며, 존재하지 않으면 오류로 취급합니다. 현재는 어린이날만 있습니다.
8. D-day 설정 기능으로 인해 세이브파일이 호환되지 않습니다.
9. D-day 설정 기능으로 인해 Human-Readable 세이브 파일의 형식이 바뀌었습니다.
10. 랜덤 레코드 생성기를 업데이트 하여, 그럴듯한 제목과 텍스트를 생성하며, 샘플 개수도 20개로 늘었습니다.
11. void printColorStrip(int colorNum) 을 통해 0, 1, 2, 3(흰색 스트립, 녹색, 청색, 보라색)의 컬러 스티커를 출력 할 수 있도록 하였습니다.
12. 9999 시간에 대해 All Day Long 텍스트의 출력 및 최우선 정렬이 가능합니다.
13. (버그 수정) 공휴일로 고정된 일정이 계속해서 증식하는 문제를 해결했습니다.

<별첨>



특정 조건 하에서 프로그램 실행이 멈추는 것을 확인했고, gdb 확인 결과 length = 0으로 고정된 상태에서 출력 대상 문자열이 아직 남은 상태로 무한 루프를 도는 것이 발견되었습니다.

보시다시피 공백을 기준으로 분리를 계속 시도하여 시작 인덱스가 0으로 고정 되는 것 같은데, 일단 임시방편으로, 코드 보시면 아시겠지만 문자열이 아직 남았고 length가 0이면 +1 해라 이런 식으로 고치긴 했는데, 위쪽에 문자열 토큰 기반 분리하는 부분을 한번 이번 주 모임에서 살펴봐야 할 것 같습니다.