パス文字列からディレクトリ名やファイル名を取得

Linuxでパス文字列からディレクトリ名やファイル名を分割取得するのに、各要素を解析取得する関数を作ったので備忘録を残す。

背景

Windowsでは、ファイルパスを分割するのに VC++では _splitpath(), Borland C++では splitpath() などの便利な関数が用意されている。
Linuxではディレクトリ名やファイル名を取得する dirname() や basename() 関数が使えるが、所望する結果が得られなかったり、引数のパス領域が書き換えられたりする。そこで、文字列からファイル名要素を抽出する関数を作った。

ファイル要素解析関数

以下にプログラムコードを示す。

_parsepath.h

```
/**
 * @file   _parsepath.h
 * @brief  Parse a file elements from a string function header.
 * @author T.Yokobayashi de JR4QPV
 * @date   2020/06/28
 */
#ifndef ___PARSEPATH_H__
#define ___PARSEPATH_H__

/**** Constant define ****/
#define   SZ_PATH_MAX       (256)

#define   FLG_EXTENSION     (1)
#define   FLG_FILENAME      (2)
#define   FLG_DIRNAME       (4)
#define   FLG_BASENAME      (8)
#define   FLG_WILDCARDS     (0x10)
#define   FLG_ABSPATH       (0x20)

/**** Interface define ****/
#ifdef __cplusplus
extern "C" {
#endif

/* ----- _parsepath.c ----- */
extern int _parsepath(const char *path, char **dirpos, char **basepos,
char **extpos);
```

```
#ifdef __cplusplus
}
#endif
#endif /* ___PARSEPATH_H__ */

/**
 *-----------------------------------------------------------------------
-------
 * @file   _parsepath.h
 * History
 * -------
 * - 2020/06/28 New created by JR4QPV.
 */
```

_parsepath.c

```
/**
 * @file   _parsepath.c
 * @brief  Parse a file elements from a string.
 * @author T.Yokobayashi de JR4QPV
 * @date   2020/06/28
 */

#include <string.h>         /* for strlen,strrchr */
#include "_parsepath.h"

/**
**********************************************************************
******
 * @brief  パス文字列からファイル名要素の位置取得
 * @param  path    パス文字列
 * @param  dirpos  ディレクトリ名要素開始位置の格納先(NULL時は格納しない)
 * @param  basepos ベース名要素開始位置の格納先(NULL時は格納しない)
 * @param  extpos  拡張子名要素開始位置の格納先(NULL時は格納しない)
 * @return 解析結果 =0:ファイル要素なし, ≠0:ファイル要素あり(要素フラグ)\n
 *         (FLG_DIRNAME|FLG_BASENAME|FLG_FILENAME|FLG_EXTENSION| \n
 *          FLG_WILDCARDS|FLG_ABSPATH)
 * @details
 *  - pathの文字長が不正の時は「0」を返す。
 * - ベース名要素には、ファイル名要素と拡張子要素が含まれる。
 *  - pathの内容は変更しない非破壊で、各要素の開始位置のポインタを返す。\n
 *  各要素は、解析結果のポインタ位置から抽出する。
 * - 各要素は以下で抽出できる□\n
 *    ディレクトリ名：*dirpos □ *basepos (最後は必ず'/') \n
 *    ベース名：       *basepos 以降 \n
 *    ファイル名：     *basepos □ *extpos \n
 *   拡張子□         *extpos 以降('.'で始まる)
 *
 * @attention
```

```
 * - マルチバイト文字には非対応□(UTF-8ならたぶん大丈夫)
*****************************************************************
******
*/
int _parsepath(const char *path, char **dirpos, char **basepos, char
**extpos)
{
    int n, flg;
    char *p, *dname, *bname, *ename;

    n = strlen(path);
    if ((n <= 0) || (n >= SZ_PATH_MAX))
        return 0;                               /* 文字列長が不正 */

    dname = bname = (char*)path;
    ename = (char*)&path[n];
    flg = 0;

    /* ディレクトリ名の有無チェック */
    if ((p = strrchr(path, '/')) != NULL) {
        /* dirname要素あり */
        bname = p+1;
        flg |= FLG_DIRNAME;

        /* 絶対パス名かチェック */
        if (dname[0] == '/')                    /* '/'で始まるか？ */
            flg |= FLG_ABSPATH;
    }

    /* ベース名の有無チェック */
    n = strlen(bname);
    if (n > 0) {
        /* basename要素あり */
        flg |= FLG_BASENAME;

        /* 拡張子の有無チェック */
        if ((p = strrchr(bname, '.')) != NULL) {
            /* ext要素あり */
            ename = p;
            flg |= FLG_EXTENSION;

            /* ファイル名の有無チェック */
            if (bname != ename)
                flg |= FLG_FILENAME;
        }

        /* ワイルドカードの有無チェック */
        p = bname;
        while (*p != '\0') {
            if ((*p == '*') || (*p == '?')) {
                flg |= FLG_WILDCARDS;           /* ワイルドカード文字を含む */
```

```
                break;
            }
            p++;
        }
    }

    /* チェック結果の格納 */
    if (dirpos != NULL) *dirpos = dname;     /* ディレクトリ名開始位置 */
    if (basepos != NULL) *basepos = bname;     /* ベース名開始位置 */
    if (extpos != NULL) *extpos = ename;     /* 拡張子開始位置 */

    return flg;
}

/**
 *-----------------------------------------------------------------------
-------
 * @file   _parsepath.c
 * History
 * -------
 * - 2020/06/28 New created by JR4QPV.
 */
```

本関数の特徴は、入力した文字列を解析して、各要素の開始ポインタ位置を戻す。又、戻り値でどの要素が存在してるかを知ることができる。又、入力した文字列は参照のみで非破壊なので、不要なコピー領域をmallocなどで確保しておかなくても必要な要素のみを抽出できる。

使い方

上記関数を使ったサンプルプログラム。

ディレクトリ名とベース名を取得

sample1.c

```
#include <stdio.h>
#include <string.h>
#include "_parsepath.h"

int main(int argc, char *argv[])
{
  char *inpath = "/home/user/test.txt";
  char *dname, *bname;
  char dir[SZ_PATH_MAX], base[SZ_PATH_MAX];
  int flg;
```

```
  flg = _parsepath(inpath, &dname, &bname, NULL);
  if (flg & FLG_DIRECTORY) {            /* dir要素あり？*/
    int n = bname - dname;              /* dir文字数(最後の'/'含む) */
    strncpy(dir, dname, n);
    dir[n] = '\0';
  }
  else {                                /* dir要素なし */
    strcpy(dir, "");
  }

  if (flg & FLG_BASENAME) {             /* base要素あり？*/
    strcpy(base, bname);
  }
  else {                                /* base要素なし */
    strcpy(base, "");
  }

  printf("dir=%s, base=%s\n", dir, base);   /* "/home/user/",
"test.txt" */

  return 0;
}
```

コンパイル実行

上の3つのソースファイルを任意のフォルダにダウンロードし、コンパイル&実行してみる。

```
$ gcc sample1.c _parsepath.c -o sample

$ ./sample
dir=/home/user/, base=test.txt
```

参考

---

1. _splitpath - Microsoft Docs 2. Man page of BASENAME

- - - -

From:
https://yokoweb.net/dokuwiki/ - **新石器Wiki**

Permanent link:
https://yokoweb.net/dokuwiki/programing/linux-programing/file-splitpath-parse?rev=1593323241

Last update: **2020/06/28 14:47**