C 言語 022 日付、時刻

【UNIX 時間】

「1970 年 1 月 1 日 0 時 0 分 0 秒」から「現在の日時」までの経過秒数を「UNIX 時間」と称し、C 言語では日時を UNIX 時間で管理しています。

【time_t 型】

UNIX 時間を記録するのに使用するデータ型です。

「time.h」に「typedef int time_t;」で宣言されています。

【time 関数】

定　　義	time_t time(time_t* timer);
ﾍｯﾀﾞｰﾌｧｲﾙ	time.h
説　　明	UNIX 時間を取得する。
戻 り 値	正常：UNIX 時間／異常：-1
例　　文	time_t now = time(&now);　　/* 現在の日時を秒数で取得 */

time 関数は、システム／OS から UNIX 時間を取得する関数です。

戻り値も参照渡しの引数も同じ値になる仕様なので、下記の書き方でも変数 now は同じ値になります。

A)　time(&now);

B)　now = time(NULL);

【ctime 関数】

定　　義	char* ctime(const time_t* timer);
ﾍｯﾀﾞｰﾌｧｲﾙ	time.h
説　　明	timer から「曜日 月 日 時:分:秒 西暦年」形式に変換した文字列を返す関数。 改行が含まれるので注意。
戻 り 値	正常：文字列／異常：NULL
例　　文	time_t now = time(NULL); printf("%s", ctime(&now));　　/* ¥n は不要 */

【tm 型】

tm 型は「time.h」に宣言されている構造体で、日時データを記録するのに使用します。
メンバ変数の定義は下記の通りです。

```
struct tm{
    int tm_sec;      /* 秒、0～61（最大 2 秒までのうるう秒を考慮している）
    int tm_min;      /* 分、0～59
    int tm_hour;     /* 時、0～23
    int tm_mday;     /* 日、1～31
    int tm_mon;      /* 月、0～11（0 が 1 月）
    int tm_year;     /* 年、1900 からの経過年数
    int tm_wday;     /* 曜日、0～6（0 が日曜日）
    int tm_yday;     /* 0～365（年内の通し日数）
    int tm_isdst;    /* サマータイムの有無。1 以上=有効、0 以下=無効
};
```

【localtime 関数】

定　　義	struct tm* localtime(const time_t* timer);
ﾍｯﾀﾞｰﾌｧｲﾙ	time.h
説　　明	timer の UNIX 時間を tm 型に変換する関数。
戻 り 値	正常：tm 型のアドレス／異常：NULL

＜例文＞

```
time_t now = time(NULL);                /* UNIX 時間を取得 */
struct tm* tmdata = localtime(&now);    /* tm 型に変換 */
char* weak[] = {"日", "月", "火", "水", "木", "金", "土"};

/* メンバ変数を出力 */
printf("%d 年", tmdata->tm_year + 1900);
printf("%d 月", tmdata->tm_mon + 1);
printf("%d 日 ", tmdata->tm_mday);
printf("%s 曜日 ", weak[tmdata->tm_wday]);
printf("%d 時", tmdata->tm_hour);
printf("%d 分", tmdata->tm_min);
printf("%d 秒 ", tmdata->tm_sec);
printf("今年%d 日目", tmdata->tm_yday);
```

【asctime 関数】

定　　義	char* asctime(const struct tm* tm_data);
ﾍｯﾀﾞｰﾌｧｲﾙ	time.h
説　　明	tm_data から「曜日 月 日 時:分:秒 西暦年」形式に変換した文字列を返す関数。
戻 り 値	正常：文字列／異常：NULL
例　　文	time_t now = time(NULL);　　/* time_t 型の変数 */ struct tm* tm_data = localtime(&now);　/* tm 型に変換 */ printf("%s¥n", asctime(tm_data));　　/* 曜日 月 日 時:分:秒 西暦年で出力 */

【mktime 関数】

定　　義	time_t mktime(struct tm* tm_data);
ﾍｯﾀﾞｰﾌｧｲﾙ	time.h
説　　明	tm_data を time_t 型に変換する関数。
戻 り 値	正常：UNIX 時間／異常：-1

＜例文＞

```
struct tm tmdata;                    /* tm 型 */
time_t timedata;                     /* time_t 型 */

/* 日時を設定する */
tmdata.tm_year = 2017 - 1900;        /* 2017 年 */
tmdata.tm_mon = 2 - 1;               /* 2 月 */
tmdata.tm_mday = 14;                 /* 14 日 */
tmdata.tm_hour = 12;                 /* 12 時 */
tmdata.tm_min = 30;                  /* 30 分 */
tmdata.tm_sec = 45;                  /* 45 秒 */
tmdata.tm_isdst = 0;                 /* サマータイム無し */
tmdata.tm_wday = 0;                  /* 0 を代入 */
tmdata.tm_yday = 0;                  /* 0 を代入 */

/* tm 型から time_t 型に変換 */
timedata = mktime(&tmdata);          /* 引数がポインタなのに注意 */

printf("%d¥n", timedata);            /* UNIX 時間で出力 */
printf("%s¥n", ctime(&timedata));    /* 曜日 月 日 時:分:秒 西暦年で出力 */
```

tm 構造体のメンバ変数「tm_wday」と「tm_yday」は年月日さえ分かれば算出する事ができるので、0 で初期化しても構いません。
ただし、コンパイラによっては「asctime 関数」「mktime 関数」の引数に、メンバ変数が初期化されていない tm 構造体を指定すると変換に失敗するものもあります。

【2038 年問題】
多くのコンパイラは time_t 型を「signed long int 型」で扱っており、「2038 年 1 月 19 日 3 時 14 分 7 秒」には long 型の格納範囲を超えるため、オーバーフローを起こす問題を抱えています。