THInterpreter 메뉴얼 1.0

http://www.mearee.com/ 메아리

1. 기본 언어 관련 사항

THInterpreter 는 C 언어를 기반으로 하는 명령어 해석기입니다. 기본적으로 C 언어를 사용할 수 있으면, 한 시간 안에 THInterpreter를 이용하여 프로그램을 만들 수 있도록 유사하게 만들었습니다.

다만, 차이점이 몇가지 있습니다.

(1) 포인터 자료형이 없습니다. 예를 들어 C언어에서 문자열을 저장하기 위해 char * 포인터 형을 선언하여 사용 할 수 있지만, THInterpreter에서는 그냥 string 이라는 데이터 타입을 사용해서 스트링 문자를 표현 합니다.(Java의 String 처럼)

(2) Memory reference 호출이 없습니다. 즉, C언어에서는 함수를 호출 할 때, 전달인자로 call by reference 형식 (&a, &b 같은) 으로 데이터를 입력 하거나, 결과 값을 받아 올 수 있는데, THInterpreter에서는 전달인자는 call by value 형식 입니다.

(3) handle 데이터 타입은 C언어에서의 void * 형태와 같은 의미 입니다. 즉, 내부적으로 여러 구조체나 데이터 변수를 포인터로 선언한 것을 리턴 받는 것입니다.(handle 데이터 타입 설명 참조)

(4) string 데이터 타입은 문자열 뿐만 아니라 여러 형태의 데이터를 저장 할 수 있습니다. 내부적으로는 unsigned char * 형의 데이터로 관리 합니다. (string 데이터 타입 설명 참조)

모든 프로그램은 main() 함수에서 시작합니다.

2. 내장 명령어

(1) if else

(2) while

(3) for

(4) return

(5) break

(6) continue

(7) 기타

3. 기본 데이터 타입

(1) void

(2) string

(3) char

(4) byte

(5) int/int8 / int16 / int32 / int64

(6) uint8 / uint16 / uint32 / uint64

(7) float / float32 / float64

(8) bigint

(9) bigfloat

(10) handle

(11) 기타

4. 기본 입출력 함수

(1) int32 printf(IN string format, ...)

설명: 지정된 포맷 스트링 형식으로 표준 출력을 통해 스트링을 출력 한다.

포맷 형식은 다음 문자열을 사용 할 수 있음.

a. %% : 화면에 % 문자를 찍는다.

b. %d : 숫자를 출력 시킨다.(int32 데이터 출력용)

c. %D : 모든 종류의 숫자를 출력 시킨다. (bigint 데이터 출력용)

d. %s : 문자열을 출력 시킨다

e. %c : 하나의 문자를 출력 시킨다.

f. %x : 변수의 값을 헥사 값으로 출력 시킨다

g. %b : 변수의 값을 2진수로 출력 시킨다.

h. %f : 실수값을 출력 한다.

i. ₩xHH : 헥사값 0xHH 에 해당하는 문자를 출력 시킨다

j. ₩NNN : 10진수 숫자 NNN에 해당 하는 문자를 출력 시킨다.

k. ₩0nn : 8진수 nn에 해당하는 문자를 출력 시킨다. (숫자가 0부터 시작함)

반환값: 출력된 문자의 개수

(2) int32 sprintf(OUT string strbuff, IN string format, ...)

설명: 지정된 포맷(format)으로 문자열 변수(strbuff)에 출력을 저장한다.

반환값: 출력된 문자의 개수

(3) int32 puts(IN string strbuff)

설명: 문자열(strbuff)를 표준 출력을 통하여 출력한다.

반환값: 출력된 문자 개수

(4) int32 scanf(IN string format, ...)

설명: 표준 입력을 통하여 변수의 값을 입력 받는다.

반환값:입력 받은 문자 개수

(5) int32 strlen(IN string strbuff)

설명: 지정된 문자열(strbuff)의 길이를 반환 한다. (스트링 길이.. NULL 문자가 있을때 까지)

반환값: 지정된 문자열(strbuff)의 길이

(6) int32 strcmp(IN string strbuff1, IN string strbuff2)

설명: 지정된 두개의 문자열을 비교 한다.

반환값: 0이면 두개의 문자열은 동일함. 아니면, 서로 다른 문자열임

(7) int32 strcasecmp(IN string strbuff1, IN string strbuff2)

설명: 대소문자 구분 하지 않고 문자열을 비교한다

반환값: 0이면 두개의 문자열은 동일함. 아니면, 서로 다른 문자열임

(8) int32 strncmp(IN string strbuff1, IN string strbuff2, IN int32 len)

설명: 두개의 문자열을 처음부터 len 개수 만큼만 비교한다.

반환값: 0이면 두개의 문자열은 len 개수까지는 동일 하다. 0이 아니라면, 다른 문자열임.

(9) int32 strncasecmp(IN string strbuff1, IN string strbuff2, IN int32 len)

설명: 두개의 문자열을 처음부터 len 개수까지 대소문자 구분 없이 비교 한다.

반환값: 0이면 두개의 문자열은 대소문자 구분 없이 len개 까지 동일함. 0이 아니라면, 다른 문자열임.

(10) int32 strcpy(IN string deststr, IN string srcstr)

설명: 원본 문자열(srcstr)을 대상 문자 변수(deststr)에 복사한다.

반환값: 복사된 문자열의 개수

(11) int32 strncpy(IN string deststr, IN string srcstr, IN int32 len)

설명: 원본 문자열(srcstr)의 처음부터 len개 문자를 저장 변수(deststr)에 복사한다.

반환값: 복사된 문자열(deststr)의 길이

(12) int32 strcat(IN string str1, IN string str2)

설명: 문자열(str1)의 끝에 새로운 문자열(str2)를 복사한다

반환값: 연결된 후의 str1 문자열의 길이

(13) int32 strncat(IN string str1, IN string str2, IN int32 len)

설명: 문자열(str1)의 끝에 새로운 문자열(str2)를 len 길이 만큼만 복사한다.

반환값: 연결된 후의 str1 문자열의 길이

(14) int32 tolower(IN char ch)

설명: 문자(ch)를 소문자로 변환 하여 반환 한다.

반환값: 문자(ch)가 영문자의 경우 소문자에 해당하는 아스키 코드 값, 영문자가 아닌 경우에는 0 반환

(15) int32 toupper(IN char ch)

설명: 문자(ch)를 대문자로 변환 하여 반환 한다.

반환값: 문자(ch)가 영문자의 경우 대문자에 해당하는 아스키 코드 값, 영문자가 아닌 경우에는 0 반환

(16) int32 getchar(void)

설명: 표준 입력을 통하여 하나의 문자를 입력 받는다.

반환값: 표준 입력을 통해 입력 받은 문자의 아스키 코드

(17) int32 isalpha(IN char ch)

설명: 문자(ch)가 알파벳인가 평가 한다.

반환값: 만약, 알파벳 문자라면 문자에 맞는 아스키 코드 값을 반환 한다. 아니라면, 0을 반환 한다.

(18) int32 isalnum(IN char ch)

설명: 문자(ch)가 알파벳이나 숫자 문자인가 평가 한다

반환값: 만약, 알파벳이나 숫자라면 해당 문자의 아스키 코드 값을 반환 한다. 아니라면 0을 반환 한다.

(19) int32 isascii(IN char ch)

설명: 문자(ch)가 아스키 문자인지 평가 한다.

반환값: 만약, 아스키 문자라면 해당 문자의 아스키 코드 값을 반환 한다. 아니라면 0을 반환 한다.

(20) int32 iscntrl(IN char ch)

설명: 문자(ch)가 컨트롤 문자인지 평가 한다.

반환값: 만약, 컨트롤 문자라면, 해당 문자의 아스키 코드 값을 반환 한다. 아니라면 0을 반환 한다.

(21) int32 isdigit(IN char ch)

설명: 문자(ch)가 숫자인지 평가한다.

반환값: 만약, 숫자 문자라면 해당 문자의 아스키 코드 값을 반환 한다. 아니라면 0을 반환 한다.

(22) string getenv(IN string name)

설명: 환경 변수(name)에 해당하는 값을 획득 한다.

반환값: 환경 변수값의 문자열

(23) int32 setenv(IN string name, IN string value)

설명: 환경 변수(name)에 새로운 값(value)을 설정 한다.

반환값: 성공하면 0을 반환, 실패하면 -1을 반환 한다.

(24) int32 isxdigit(IN char ch)

설명: 입력 문자(ch)가 16진수 문자인가 평가한다.

반환값: 16진수 문자라면 해당 문자의 아스키 코드 값을 반환. 아니라면 0을 반환 한다.

(25) int32 atoi(string str) / int64 atoll(string str)

설명: 입력 문자열(str)을 숫자(int32, int64)으로 해석하여 그 값을 반환 한다.

반환값: 0 이상의 값. (str이 숫자로 이루어진 문자열이 아니라면 0 반환)

(26) uint32 time(void)

설명: 현재 시간 값(초)을 획득 한다.

반환값: 현재 시간 값

(27) int32 localtime_year(IN uint32 time)

설명: 현재 시간 값(time)을 이용하여 1900 년 이후의 연도를 획득 한다.(year -1900)

반환값: 시간에 해당하는 연도 값

(28) int32 localtime_mon(IN uint32 time)

설명: 현재 시간 값(time)을 이용하여 현재 해당하는 월을 획득 한다. (0 - 11)

반환값: 시간에 해당하는 월 값

(29) int32 localtime_mday(IN uint32 time)

설명: 현재 시간 값(time)을 이용하여 해당하는 날을 획득 한다. (1-31)

반환값: 시간에 해당하는 일 값

(30) int32 localtime_hour(IN uint32 time)

설명: 현재 시간 값(time)을 이용하여 해당하는 시간을 획득 한다. (0-23)

반환값: 시간에 해당하는 시각 값.

(31) int32 localtime_min(IN uint32 time)

설명: 현재 시간 값(time)을 이용하여 해당하는 분 값을 획득 한다. (0-59)

반환값: 시간에 해당하는 분 값

(32) int32 localtime_sec(IN uint32 time)

설명: 현재 시간 값(time)을 이용하여 해당하는 초 값을 획득 한다. (0-59)

반환값: 시간에 해당하는 초 값

(33) int32 localtime_wday(IN uint32 time)

설명: 현재 시간 값(time)을 이용하여 해당하는 요일 값을 획득한다. (0-6)

반환값: 시간에 해당하는 요일 값

(34) int32 localtime_yday(IN uint32 time)

설명: 현재 시간 값(time)을 이용하여 해당하는 날 수를 획득 한다. (0-365)

반환값: 일년중 시간에 해당하는 날 수

(35) int32 rand(void)

설명: 난수를 발생시켜 값을 반환 한다.

반환값: 불규칙적 숫자

(36) void srand(IN int32 seed)

설명: 난수 발생의 기초가 되는 시드값을 설정한다

(37) string getenv(IN string name)

설명: 지정된 이름(name)의 환경 변수 값을 획득 한다.

반환값: 환경 변수 스트링

(38) int32 setenv(IN string name, IN string value)

설명: 지정된 이름(name)으로 새로운 환경 변수 값(value)을 설정 한다.

반환값: 성공하면 0을 반환, 아니면 -1 반환

(39) int32 handletype(IN handle hvalue)

설명: 지정된 핸들(hvalue)의 타입을 획득 한다.

반환값: 다음 값들중 하나의 값을 가진다.

HANDLE_TYPE_ERROR	에러 발생
HANDLE_TYPE_THIMAGE	이미지 제어 핸들
HANDLE_TYPE_THCRYPT	암호 제어 핸들
HANDLE_TYPE_THCRYPT_ALG	암호 제어(알고리즘) 핸들
HANDLE_TYPE_FILEIO	파일 입출력 핸들
HANDLE_TYPE_FILEIO_DIR	디렉토리 검색 핸들
HANDLE_TYPE_FILEIO_ENTRY	디렉토리 엔트리 핸들
HANDLE_TYPE_POPEN	파이프 입출력 핸들
HANDLE_TYPE_PLUGIN	플러그인 제어 핸들
HANDLE_TYPE_FDSET	소켓 모음 핸들
HANDLE_TYPE_FTPCLIENT	FTP 접속 클라이언트 핸들
HANDLE_TYPE_HTTPCLIENT	HTTP 접속 클라이언트 핸들
HANDLE_TYPE_PATTERN	정규식 패턴 핸들
HANDLE_TYPE_THREAD	쓰레드 핸들
HANDLE_TYPE_ARCHIVE	아카이브 핸들
HANDLE_TYPE_ARCHIVE_ENTRY	아카이브 파일 엔트리 핸들
HANDLE_TYPE_ARCHIVE_FILE	아카이브 파일 접근 핸들

(40) void exit(IN int32 exitCode)

설명: 프로그램 실행을 종료 한다. 종료 코드는 지정된 값(exitCode)를 발생 시킨다.

5. 문자열 함수

(1) int32 THStr_IndexOf(IN string strbuff, IN char delimiter, IN int32 offset)

설명: 문자열 버퍼(strbuff)에서 offset이후에 나오는 첫번째 문자(delimiter)의 위치를 반환 한다.

반환값: 0 이상일 경우, 해당 오프셋 번지에 처음 delimiter 문자가 존재 함.

마이너스 값일 경우, 해당 문자가 발견되지 않음.

(2) int32 THStr_LastIndexOf(IN string strbuff, IN char delimiter, IN int32 offset)

설명: 문자열 버퍼(strbuff)에서 뒤쪽에서 부터 검색하여 offset 이후에 나오는 첫번째 문자(delimiter)의 위치를 반환 한다.

반환값: 0 이상일 경우, 마지막 발견된 위치, 마이너스의 경우, 해당 문자가 발견되지 않음.

(3) bigint THStr_StringToInteger(IN string strbuff)

설명: 문자열 버퍼(strbuff)를 숫자형 데이터로 변환 한다. 숫자 데이터는 기본적으로 bigint 타입이지만, int32 혹은 int64 데이터 형으로 받아도 됨

반환값: 문자열을 해석(10진수)해서 숫자 값으로 반환함

(4) bigfloat THStr_StringToFloat(IN string strbuff)

설명: 문자열 버퍼(strbuff)를 숫자형 데이터로 변환 한다. 숫자는 bigfloat 형태의 실수 값

반환값: 문자열 해석(10진수)해서 실수 값으로 반환함

(5) int32 THStr_Substring(OUT string substr, IN string strbuff, IN int32 startoffset, IN int32 endoffset)

설명: 문자열 버퍼(strbuff)의 시작 오프셋(startoffset)과 끝 오프셋(endoffset) 사이의 문자열을 잘라내어 스트링 변수(substr)에 저장한다.

만약, endoffset이 -1 이라면, 시작 오프셋(startoffset)에서 부터 끝까지 복사한다.

반환값: substr에 저장되는 문자열의 길이

(6) int32 THStr_StringReplace(OUT string outstr, IN string instr, IN string oldstr, IN string newstr, IN byte bAll, IN byte bIgnoreCase)

설명: 원본 스트링(instr)에 존재하는 문자열(oldstr)을 새로운 문자열(newstr)로 교체하여, 새로운 문자열 변수(outstr)에 저장한다.

bAll 값이 0이면 원본 스트링(instr)의 처음부터 검색해서 처음 발견되는 문자열(oldstr)을 단 한번만 새로운 문자열(newstr)로 교체하여 새로운 문자 버퍼에 저장한다. 만약, bAll 값이 FALSE(0)이 아니라면 발견되는 모든 문자열(oldstr)에 대해 새로운 문자열(newstr)로 교체 하여 저장한다.

bIgnoreCase 값이 FALSE(0)이면, 기존의 문자열(oldstr)을 검색하는데 대소문자 구분을 한다. 만약, bIgnoreCase가 0이 아니라면 대소문자 구분 없이 모든 문자열을 교체한다.

반환값: 기존의 문자열(oldstr)이 새로운 문자열(newstr)로 교체된 전체 횟수

(7) byte THStr_StartsWith(IN string strbuff, IN string findstr, IN byte bIgnoreCase)

설명: 문자열 버퍼(strbuff)가 지정된 문자열(findstr)로 시작되는지 검사하여, 결과를 반환 한다.

bIgnoreCase값이 FALSE(0)이 아닐경우 시작 문자열을 비교하는데 대소문자 구분을 하지 않는다.

반환값: FALSE(0) 이면 시작 문자열이 아님. 0이 아니라면 시작 문자열이 일치한다.

(8) void THStr_SplitPath(IN string fullpath, OUT string directory, OUT string filename, OUT string extension)

설명: 문자열로 지정된 전체 경로(fullpath)를 디렉토리(directory)와 파일명(filename), 확장자(extension) 문자열로 나누어 저장한다.

반환값: 없음

(9) int32 THStr_Trim(OUT string deststr, IN string srcstr)

설명: 원본 문자열(srcstr)에 존재하는 앞뒤 공백문자를 제거하여 새로운 문자열 변수(deststr)에 저장한다.

반환값: 새로운 문자열 변수(deststr)에 저장된 문자열 길이

(10) int32 THStr_TokenValue(OUT string valstr, IN string srcstr, IN string tokenstr, IN byte btAssign, IN byte btDelimiter)

설명: 원본 문자열(srcstr)에서 토큰 문자열(tokenstr)에 할당된 값을 변수(valstr)에 저장한다. btAssign은 토큰 저장 문자를 지정하는 것으로 일반적으로 '=' 문자를 설정함.

btDelimiter는 원본 문자열(srcstr)에 저장된 tokenstr = value 문장을 구분하는 구분자로 콤마(,)로 구분하기 위해서는 ‘,’를 설정해야함.

반환값: 0 이상이면 토큰 문자열에 해당하는 값 문자열의 길이, -1이면 해당 토큰을 찾을 수 없음

(11) int32 THStr_TokenGet(OUT string token, IN string srcstr, IN int32 nitem, byte btDelimiter, byte blgnoreDup)

설명: 원본 문자열(srcstr)을 구분 문자열(btDelimiter)로 구분하여 나누었을 경우, 지정된 번호(nitem >= 0) 인덱스에 있는 문자열을 새로운 변수(token)에 저장한다. 스트링 구분 인덱스 번호는 0 이 첫번째 문자열임.

만약, blgnoreDup 값이 FALSE(0)이 아니라면, 문장의 시적점에 존재하는 btDelimiter 문자열을 무시한다.

반환값: 새로 저장되는 문자열(token)의 길이, 저장시킬 문자열이 없으면 -1을 반환 한다.

(12) int32 THStr_TokenCount(IN string strbuff, IN byte btDelimiter, IN byte blgDup)

설명: 문자열(strbuff)를 구분문자(btDelimiter)로 구분하였을 경우 몇개의 문자열로 나누어지는지 반환 한다. blgDup값이 0이 아닐 경우, 문자열(strbuff)의 시작지점에 있는 구분문자(btDelimiter)를 무시 한다.

반환값: 구분 문자(btDelimiter)로 나누어지는 문자열의 개수

(13) int32 THStr_FindKeyword(IN string strbuff, IN int32 nstrbuff, IN string keyword, IN int32 nkeyword)

설명: 문자열(strbuff)에서 키워드(keyword) 문자열의 시작점을 반환 한다. nstrbuff와 nkeyword는 문자열의 길이와 키워드의 길이를 입력한다.

반환값: 0이상일 경우, 문자열(strbuff)에서 키워드(keyword) 문자열이 시작되는 위치

(14) void THStr_GetTokenValue(IN string strbuff, IN int32 nstrbuff, IN string keyword, IN int32 nkeyword, OUT string valstr)

설명: 문자열(strbuff)에서 지정된 키워드 문자열(keyword) = 값 형식으로 할당된 문자열 값을 valstr 변수에 저장 한다.

반환값: 없음

(15) void THStr_ConvertIntoToCommaInt(IN bigint biVar, OUT string commastr)

설명: 숫자 값(biVar)을 3자리수 마다 콤마(,)가 있는 문자열로 변환 하여 스트링 변수(commastr)에 저장한다.

(16) byte THStr_CharAt(IN string str, IN int32 index)

설명: 문자열 버퍼(str)에서 index번째 문자를 반환 한다.

반환값: index값이 str 문자열 범위내에 있으면 해당 문자값, 범위를 벗어나면 0 을 반환 한다.

(17) void THStr_SetCharAt(IN string str, IN int32 index, byte btChar)

설명: 문자열 버퍼(str)의 index번째 문자를 btChar로 설정 한다.

(18) string THStr_Utf8ToUtf16(IN string str, IN int32 strlen)

설명: Utf8 인코딩된 문자열(str)을 지정된 길이(strlen)만큼 UTF-16 인코딩으로 변환 한다.

반환값: UTF16 문자열

(19) string THStr_Utf16ToUtf8(IN string str, IN int32 strlen)

설명: Utf16 인코딩된 문자열(str)을 지정된 길이(strlen)만큼 UTF-8 인코딩으로 변환 한다.

반환값: UTF8 문자열

(20) byte THStr_IsUtf8(IN string str, IN int32 strlen)

설명: 지정된 문자열(str)이 Utf8 문자열인지 판단한다.

반환값: UTF-8 문자열이라면 TRUE를 반환, 아니라면 FALSE 반환

(21) string THStr_UUDecode(IN string str, IN int32 strlen)

설명: UUEncoding 된 문자열(str)을 디코딩한다

반환값: 디코딩된 문자열 버퍼

(22) string THStr_BASE64Encode(IN string str, IN int32 strlen)

설명: 문자열버퍼(str)의 내용을 base 64로 인코딩 한다

반환값: base64인코딩된 문자열 버퍼

(23) string THStr_BASE64Decode(IN string str, IN int32 strlen)

설명: base64 인코딩된 문자열을 디코딩 한다

반환값: 디코딩된 문자열 버퍼

(24) string THStr_URLDecode(IN string str, IN int32 strlen)

설명: 인코딩된 URL 문자열을 디코딩 한다

반환값: 디코딩된 문자열 버퍼

(25) int32 THStr_DetectEncoding(IN string str, IN int32 strlen)

설명: 문자열(str)의 인코딩 타입을 반환 한다

반환값: 다음 값들중 하나이상의 값이 OR 연산되어 반환 된다.

TH_ENCODING_TYPE_BINARY	바이너리 인코딩
TH_ENCODING_TYPE_UTF8	UTF8 인코딩
TH_ENCODING_TYPE_UTF16BE	UTF16 Big Endian 인코딩
TH_ENCODING_TYPE_UTF16LE	UTF16 Little Endian 인코딩
TH_ENCODING_TYPE_UTF32BE	UTF32 Big Endian 인코딩
TH_ENCODING_TYPE_UTF32LE	UTF32 Little Endian 인코딩
TH_ENCODING_TYPE_TEXT	ASCII 텍스트
TH_ENCODING_TYPE_HANGUL	한글 포함된 인코딩
TH_ENCODING_TYPE_NOT_UNICODE	유니코드가 아님

(26) int32 THStr_BytesToHexString(OUT string hexbuff, IN string bytesbuff, IN int32 bufflen, IN byte bCapital)

설명: 지정된 문자열 버퍼(bytesbuff)를 지정된 길이(bufflen)만큼 16진수 핵사 스트링으로 만들어 버퍼(hexbuff)에 저장한다. bCapital이 TRUE면 16진수는 대문자로, FALSE는 소문자로 저장한다.

반환값: 변환된 길이 (통상 bufflen * 2)

(27) int32 THStr_HexStringToBytes(OUT string bytesbuff, IN string hexbuff, IN int32 hexlen)

설명: 지정된 헥사 스트링 버퍼(hexbuff)를 바이트 문자열(bytesbuff)로 변환한다.

반환값: 변환될 길이 (일반적으로 hexlen / 2)

(28)

6. 파일 입출력 함수

(1) handle fopen(IN string filename, IN int32 mode)

설명: 표준 입출력을 통하여 파일(filename)을 지정된 모드(mode)로 연다.

mode에 들어갈 수 있는 값은 다음과 같다.

FILE_OPEN_READ : 파일을 읽기 전용으로 연다. 파일이 존재 하지 않으면 실패.

FILE_OPEN_WRITE : 파일을 쓰기 전용으로 연다. 파일이 존재 하지 않으면 실패

FILE_OPEN_APPEND : 파일 끝에 추가하는 방식으로 연다. 파일이 존재 하지 않으면 실패

FILE_OPEN_CREATE : 파일을 생성 하여 연다. 파일이 존재 하면 실패

FILE_OPEN_RW : 파일을 읽기, 쓰기 전용으로 연다. 파일이 존재하면 실패

반환값: 파일 포인터에 대한 핸들(handle) 값

(2) byte fisfailed(IN handle fHandle)

설명: 지정된 파일 핸들(fHandle)이 열기에 실패했는지 조사 한다.

반환값: TRUE(1)라면 실패했음. FALSE(0)라면 열기에 성공 함.

(3) byte fclose(IN handle fHandle)

설명: 파일 핸들(fHandle)을 닫는다.

반환값: TRUE(1)면 닫기 성공, FALSE(0)면 실패

(4) int32 fread(IN handle fHandle, OUT string buff, IN int32 len)

설명: 파일 핸들(fHandle)로 열린 파일에 지정된 길이(len)만큼만 읽어서 데이터(buff) 버퍼에 저장한다.

반환값: 읽은 데이터 길이 반환. 만약, 음수(-)라면 파일의 끝(EOF) 이거나 실패

(5) int32 fwrite(IN handle fHandle, IN string buff, IN int32 len)

설명: 파일 핸들(fHandle)로 열린 파일에 대해 데이터 버퍼(buff)의 내용을 지정된 길이(len)만큼만 쓴다.

반환값: 쓰기 성공한 길이 반환. 만약, 음수(-)라면 실패

(6) int64 flength(IN handle fHandle)

설명: 파일 핸들(fHandle)로 지정된 열린 파일의 길이를 반환 한다.

반환값: 파일의 전체 길이

(7) int64 fseek(IN handle fHandle, IN int64 offset, IN int32 method)

설명: 파일 핸들(fHandle)로 열린 파일의 현재 위치를 새로운 오프셋(offset)으로 지정 한다.

method의 값은 다음과 같다.

FILE_SEEK_SET : 파일 포인터를 지정된 오프셋(offset)으로 정한다

FILE_SEEK_CUR : 파일 포인터를 현재 위치 + 지정된 오프셋 값으로 정한다.

FILE_SEEK_END : 파일 포인터를 끝에서부터 설정한다. 위치 값은 파일의 끝 위치 + 오프셋(offset)으로 정한다.(보통 이런 경우에는 offset값이 음수(-)를 가진다.)

반환값: 새롭게 바뀐 현재 위치

(8) int64 ftell(IN handle fHandle)

설명: 현재의 파일 포인터 위치를 반환 한다. fseek(handle, 0, FILE_SEEK_CUR)와 완전히 동일하다.

반환값: 현재의 위치 포인터

(9) int32 fprintf(IN handle fHandle, IN string format, ...)

설명: 파일 핸들(fHandle)에 지정된 포맷 형식으로 데이터를 쓴다.

반환값: 씌여진 데이터 길이

(10) int32 freadline(IN handle fHandle, OUT string buff, IN int32 len)

설명: 열린 파일 핸들(fHandle)에서 최대 지정된 길이(len) 만큼의 한 라인을 입력 받아 버퍼(buff)에 저장한다. 한 라인의 끝은 개행문자(₩n) 까지를 말하며, 개행 문자 자체는 버퍼(buff)에 저장되지 않는다.

반환값: 읽어들인 한 라인의 길이

(11) handle dopen(IN string pattern)

설명: 디렉토리 파일 검색을 위해 표준 입출력 핸들을 획득 한다. 지정된 파일 형식(pattern)을 찾기 위한 핸들.

반환값: 디렉토리 검색 핸들(handle)

예제: - 디렉토리 검색 예제

```
void test(void)
{
  handle dHandle;
  handle dEntry;
  byte bRet;

  dHandle = dopen( “/etc/*.conf” );  //   /etc 디렉토리에서 *.conf 파일을 찾기 위해 핸들을 얻는다
  if(disfailed(dHandle))
```

```
        printf( "디렉토리 열기 실패₩r₩n" );

    while(TRUE)
    {
        bRet = dnextentry(dHandle, dEntry);
        if(bRet == FALSE) break;
        printf( "Filename: %s₩n" , dentryfilename(dEntry));
        dentryclose(dEntry);
    }
    dclose(dHandle);
}
```

(12) byte disfailed(IN handle dHandle)

설명: 지정된 디렉토리 핸들(dHandle)값이 실패 한 것인지 체크 한다.

반환값: TRUE멷 디렉토리 열기에 실패, FALSE면 디렉토리 열기 성공

(13) byte dclose(IN handle dHandle)

설명: 지정된 핸들로 열린 디렉토리 핸들을 닫는다.

반환값: TRUE면, 닫기 성공. FALSE면 닫기 실패

(14) byte dnextentry(IN handle dHandle, OUT handle dEntry)

설명: 지정된 디렉토리 핸들에서 파일 엔트리 핸들(dEntry)를 획득 한다.

반환값: 파일 찾기 성공이면 TRUE, 실패면 FALSE 반환

(15) byte dentryclose(IN handle dEntry)

설명: 디렉토리 앤트리 핸들(dEntry)를 닫고, 메모리를 반환 한다.

반환값: 성공이면 TRUE, 실패면 FALSE를 반환

(16) string dentryfullpath(IN handle dEntry)

설명: 디렉토리 엔트리 핸들(dEntry)에서 전체 경로를 반환 받는다.

반환값: 전체 경로 문자열

(17) string dentryfilename(IN handle dEntry)

설명: 디렉토리 엔트리 핸들(dEntry)에서 파일명을 반환 받는다

반환값: 파일명 문자열

(18) int64 dentryfilesize(IN handle dEntry)

설명: 디렉토리 엔트리 핸들(dEntry)에서 파일 크기를 반환 받는다.

반환값: 파일 크기 값

(19) int64 dentryfilectime(IN handle dEntry)

설명: 디렉토리 엔트리 핸들(dEntry)에서 파일의 생성 시각을 반환 받는다.

시간 정보는 CCYYMMDDhhmmss 형식으로된 숫자값이다.

반환값: 생성 시각 숫자값(CCYYMMDDhhmmsss 형식)

(20) int64 dentryfileatime(IN handle dEntry)

설명: 디렉토리 엔트리 핸들(dEntry)에서 파일 접근 시각을 반환 받는다.

반환값: 접근 시각 숫자값(CCYYMMDDhhmmsss 형식)

(21) int64 dentryfilemtime(IN handle dEntry)

설명: 디렉토리 엔트리 핸들(dEntry)에서 파일 수정 시각을 반환 받는다.

반환값: 수정 시각 숫자값(CCYYMMDDhhmmsss 형식)

(22) uint32 dentryfiletype(IN handle dEntry)

설명: 디렉토리 엔트리 핸들(dEntry)에서 파일 타입을 반환 받는다.

반환값: 파일 타입 값

FILE_TYPE_DIR : 디렉토리 엔트리의 경우

FILE_TYPE_FILE : 파일 엔트리의 경우

FILE_TYPE_UNK : 그 밖의 특수 파일들

(23) byte mkdir(IN string path)

설명: 지정된 경로로 새로운 디렉토리를 생성 한다.

반환값: 성공이면 0, 실패면 -1 반환

(24) byte rmdir(IN string path)

설명: 지정된 경로의 디렉토리를 삭제 한다. (삭제전에 디렉토리가 비워져 있어야 함)

반환값: 성공이면 0, 실패면 -1 반환

(25) byte unlink(IN string path)

설명: 지정된 경로의 파일을 삭제 한다.

반환값: 성공이면 0, 실패면 -1 반환

(26) handle stat(IN string filepath)

설명: 지정된 경로의 파일에 대한 디렉토리 엔트리 핸들을 획득한다.

반환값: dentryfilename() 등의 함수에서 사용하는 디렉토리 엔트리 핸들

(27) byte chdir(IN string path)

설명: 지정된 디렉토리(path)로 현재 디렉토리를 변경 한다.

반환값: 성공하면 TRUE, 실패하면 -1 반환

(28)

7. 메모리 함수

(1) void memset(OUT mem, IN byte btChar, IN int32 len, IN int32 offset)

설명: 변수(mem)의 오프셋(offset)부터 시작해서 지정된 길이(len)만큼 특정 문자(btChar)로 채운다.

반환값: 없음.

(2) void memcpy(OUT mem1, IN mem2, IN int32 len, IN int32 mem1offset, IN int32 mem2offset)

설명: 변수2(mem1)의 시작 위치(mem2offset)에서 부터 지정된 길이(len)만큼 변수1(mem2)의 시작 위치(mem1offset)에 복사한다.

반환값: 없음.

(3) int32 memcmp(IN mem1, IN mem2, IN int32 len, IN int32 mem1offset, IN int32 mem2offset)

설명: 두개의 변수(mem1, mem2)의 특정 위치(mem1offset, mem2offset)에서 부터 지정된 길이(len)만큼 비교해서, 내용이 똑같은지 확인 한다.

반환값: 내용이 같으면 0 반환, 다르면 0이 아닌 값을 반환 한다.

(4) string malloc(IN int32 len)

설명: 지정된 길이(len) 만큼의 메모리 공간을 할당하여 문자열 변수로 반환 한다.

반환값: 문자열 값

(5) int32 sizeof(IN mem)

설명: 지정된 변수(mem)의 실제 크기를 반환 한다. 변수의 타입마다 길이를 반환하는 내용은 다음과 같다.

a. 숫자형 변수(int8 ~ int64) : 비트수가 정해진 숫자형 변수는 메모리 차지하는 크기를 반환(비트/8 한 값)

b. bigint, bigfloat: 경우 스트링으로 변환 했을때 문자열의 길이를 반환

c. 문자열(string) : 실제 메모리상에 차지하고 있는 영역의 크기 반환. strlen()과 다른 점은 strlen은 처음부터 NULL문자까지 길이를 반환 하지만, sizeof()는 NULL 문자가 포함되어 있더라도 실제 차지하는 공간의 크기를 반환 한다.

반환값: 변수가 메모리에서 차지하는 실제 길이 반환

(6) string mempack(IN vargs...)

설명: vargs 로 나열된 변수들의 바이너리 값을 문자열로 packing 하여 반환 한다.

반환값: 나열된 변수들의 값이 저장된 버퍼

(7) int32 memunpack(IN string buff, OUT vargs....)

설명: 패킹된 buff에서 나열된 변수에 해당하는 값들을 저장한다.

반환값: 저장된 변수 개수

(8)

8. 수학 함수

(1) bigfloat log10(IN bigfloat bfVar)

설명: 밑수가 10 인 상용 로그를 구한다

반환값: bfVar에 대한 상용 로그 값

(2) bigfloat log(IN bigfloat bfVar)

설명: 자연로그를 구한다.

반환값: bfVar에 대한 자연로그 값

(3) bigfloat logbn(IN bigfloat bfVar, IN bigfloat bfBase)

설명: 밑수가 bfBase인 로그 값을 구한다.

반환값: bfVar에 대한 밑수가 bfBase인 로그값

(4) bigfloat sin_r(IN bigfloat bfRadian) / bigfloat sin(IN bigfloat bfDegree)

설명: 라디안값(bfRadian) / 각도(bfDegree) 에 대한 sin() 함수 값을 구한다.

(5) bigfloat cos_r(IN bigfloat bfRadian) / bigfloat cos(IN bigfloat bfDegree)

설명: 라디안값(bfRadian) / 각도(bfDegree) 에 대한 cos() 함수 값을 구한다.

(6) bigfloat tan_r(IN bigfloat bfRadian) / bigfloat tan(IN bigfloat bfDegree)

설명: 라디안값(bfRadian) / 각도(bfDegree) 에 대한 tan() 함수 값을 구한다.

(7) bigfloat sinh_r(IN bigfloat bfRadian) / bigfloat sinh(IN bigfloat bfDegree)

설명: 라디안값(bfRadian) / 각도(bfDegree) 에 대한 쌍곡선 sinh() 함수 값을 구한다.

(8) bigfloat cosh_r(IN bigfloat bfRadian) / bigfloat cosh(IN bigfloat bfDegree)

설명: 라디안값(bfRadian) / 각도(bfDegree) 에 대한 쌍곡선 cosh() 함수 값을 구한다.

(9) bigfloat tanh_r(IN bigfloat bfRandian) / bigfloat tanh(IN bigfloat bfDegree)

설명: 라디안값(bfRadian) / 각도(bfDegree) 에 대한 쌍곡선 tanh() 함수 값을 구한다.

(10) bigfloat acos(IN bigfloat bfRadian) / bigfloat asin(IN bigfloat bfRadian) / bigfloat atan(IN bigfloat bfRandian)

설명: 라디안값(bfRadian)에 대한 Arc cos, Arc sin, Arc tan 값을 구한다.

(11) bigfloat sqrt(IN bigfloat bfVar)

설명: bfVar에 대한 제곱근(루트) 값을 구한다.

(12) bigfloat minussqrt(IN bigfloat bfVar)

설명: bfVar에 대한 마이너스 제곱근(루트) 값을 구한다. (1/bfVar 제곱)

(13) bigfloat pow(IN bigfloat bfVar, IN bigfloat bfPvar)

설명: bfVar에 대한 bfPvar 승(제곱)을 구한다.

(14) bigfloat powe(IN bigfloat bfPvar)

설명: 자연로그 e 의 bfPvar 승을 구한다.

(15) bigfloat pow10(IN bigfloat bfPvar)

설명: 10의 bfPvar 승을 구한다.

(16) bigfloat pow3(IN bigfloat bfVar)

설명: bfVar의 3승을 구한다.

(17) bigfloat pow2(IN bigfloat bfVar)

설명: bfVar의 2승을 구한다.

(18) bigfloat gete(void)

설명: 자연로그 e 값을 구한다.

(19) bigfloat getpi(void)

설명: 원주율(pi) 값을 구한다.

(20) bigfloat ncr(IN bigfloat bfN, IN bigfloat bfR)

설명: 조합 nCr 값을 구한다. (순서에 상관없이 n 에서 r개를 뽑을 경우의 수)

(21) bigfloat npr(IN bigfloat bfN, IN bigfloat bfR)

설명: 순열 nPr 값을 구한다. (순서대로 n에서 r개를 뽑을 경우의 수)

(22) bigfloat fact(IN bigint biVar)

설명: biVar의 팩토리얼 값을 구한다.

(23) int32 matrix_add(IN int32 height, IN int32 width, IN bigfloat src1[], IN bigfloat src2[], OUT bigfloat dest[])

설명: 가로(width), 세로(height) 크기의 행렬 src1[], src2[]를 더해서 dest에 저장한다.

src1이나 src2의 크기는 width * height 이 되어야 한다. 0번 행부터 (height-1) 행까지 순서대로 0번 열의 값부터 (width-1)의 값을 저장시켜야함.

즉, 행렬 좌표 (x, y)의 값(x > 0, y > 0) 은 배열에서 (x-1) + ((y-1) * width) 의 위치에 저장된다.

x 의 값 범위는 0 <= x <= (width-1) 이고, y 의 값의 범위는 0 <= y <= (height - 1) 이 된다.

반환값: 전체 결과 값 배열의 크기

(24) int32 matrix_sub(IN int32 height, IN int32 width, IN bigfloat src1[], IN bigfloat src2[], OUT bigfloat dest[])

설명: 행렬 빼기를 해서 dest에 저장한다. 나머지 자세한 설명은 matrix_add 와 동일

(25) int32 matrix_mul(IN int32 height, IN int32 width, IN bigfloat src1[], IN bigfloat src2[], OUT bigfloat dest[])

설명: 행렬 곱을 해서 dest에 저장한다. 나머지 자세한 설명은 matrix_add 와 동일

(26) int32 matrix_div(IN int32 height, IN int32 width, IN bigfloat src1[], IN bigfloat src2[], OUT bigfloat dest[])

설명: 행렬 나누기를 해서 dest에 저장한다. 나머지 자세한 설명은 matrix_add와 동일

(27) int32 equation_1(IN bigfloat A, IN bigfloat B, IN bigfloat bfResX1)

설명: Ax + B = 0 인 일차 방정식의 X 값을 구한다.

반환값: 모든 방정식 함수는 다음과 같은 반환 값을 가진다.

MATH_EQUATION_ERROR : 방정식 에러

MATH_EQUATION_1 : 방정식을 1차 방정식으로 풀었음(실수 근 1개, bfResX1에 값이 저장)

MATH_EQUATION_2 : 방정식을 2차 방정식으로 풀었음(실수 근 2개. bfResX1, bfResX2에 서로 다른 값이 저장)

MATH_EQUATION_2_SAME_X : 방정식을 2차 방정식으로 풀었으나 중근(실수근 1개, bfResX1, bfResX2 값이 같음)

MATH_EQUATION_2_IMA_X : 방정식을 2차 방정식으로 풀었으나 허근(실수부 값은 bfResX1, 허수부는 bfResX2에 저장)

2차 방정식 허근의 경우 a + i * b 와 a - i * b 두개의 근이 나오므로, bfResX1에는 a 값이,

bfResX2에는 b 값이 저장된다. (i = 허근을 나타내는 기호 / 제곱해서 -1 이 되는 수)

MATH_EQUATION_3_REAL : 3차 방정식의 실수 값 표현식

3차 방정식의 실수값 표현 규칙은 A + i * B 로 표현됨.

ResX1RealAb, ResX2RealAb, ResX3RealAb 에는 총 3개의 실수부 값(A)가 저장됨

ResX1ImaThe, ResX2ImaThe, ResX3ImaThe 에는 총 3개의 허수부 값(B)가 저장됨

MATH_EQUATION_3_ABS : 3차 방정식의 절대값 표현식

3차 방정식의 절대값 표현 규칙은 Ab (cos(Theta) * i * sin(Theta)) 로 표현됨.

ResX1RealAb, ResX2RealAb, ResX3RealAb 에는 총 3개의 절대값(Ab)가 저장됨

ResX1ImaThe, ResX2ImaThe, ResX3ImaThe 에는 총 3개의 각도값(Theta)가 저장됨

3차 방정식은 Ab (cos(The) + i * sin(The)) = RealA + i * B 임. 즉, 절대값 표현식이나 실수값이나 같음.

9. 네트워크 함수

(1) int32 NetConnectHost(IN string hostname, IN uint16 port)

설명: 호스트(hostname)의 포트(port)로 TCP 접속을 시도한다.

반환값: 성공하면 0 이상의 소켓 번호, 실패하면 -1

(2) int32 NetTimeoutConnectHost(IN string hostname, IN uint16 port, IN int32 timeout)

설명: 호스트(hostname)의 포트(port)로 TCP 접속을 시도 한다. 단, 지정된 시간(초, timeout) 이상 기다릴 경우, 에러를 반환 한다.

반환값: timeout 이내에 접속에 성공하면 소켓 번호, 실패하면 -1

(3) void NetSetBlockMode(IN int32 socket, IN byte bSet)

설명: 지정된 소켓(socket) 연결 상태를 Non blocking 모드로 설정하거나 해제 한다.

bSet 값이 TRUE 면, Non blocking mode로 설정되고, FALSE면, blocking mode로 설정 된다.

반환값: 없음

(4) int32 NetGetMyPort(IN int32 socket)

설명: 지정된 소켓(socket)으로 연결된 상태의 내 포트 번호를 반환 받는다.

반환값: 현재 내가 사용하고 있는 TCP 포트 번호

(5) int32 NetGetDestPort(IN int32 socket)

설명: 지정된 소켓(socket)으로 연결된 상태에서 상대편의 포트 번호를 반환 받는다.

반환값: 현재 상태가 사용하고 있는 TCP 포트 번호

(6) string NetGetHostname(IN string ipaddress)

설명: 문자로 구성된 IP 주소를 이용하여 호스트 이름을 획득한다. DNS 엔트리 이름 획득

반환값: 호스트 이름의 문자열 값

(7) string NetGetIPAddress(IN string hostname)

설명: 문자열로 구성된 호스트 이름을 이용하여 문자열로 구성된 IP 주소를 획득 한다.

반환값: IP주소에 대한 문자열 값

(8) string NetGetDestIPAddress(IN int32 socket)

설명: 현재 연결된 소캣(socket) 번호를 이용하여 상대편의 IP 주소를 획득 한다.

반환값: IP주소에 대한 문자열 값

(9) int32 NetPrintf(IN int32 socket, IN string format, ...)

설명: 현재 연결된 소캣(socket)에 대해 포맷 스트링에 맞춘 형식으로 데이터를 보낸다.

반환값: 소켓에 씌여진 데이터 크기

(10) int32 NetReadline(IN int32 socket, OUT string buffer, IN int32 maxlen, IN int32 timeout)

설명: 현재 연결된 소캣(socket)에서 개행 문자(₩n)이 나올때 까지 입력 받는다. 최대 입력 문자의 개수는 maxlen 이고, 특정 시간(초, timeout) 이상 입력이 없으면 그대로 반환 한다.

반환값: 입력 받은 문자열의 길이. timeout 동안 아무런 입력이 없다면 0 반환, 네트워크가 끊기거나 에러가 발생하면 -1 반환

(11) int32 NetRead(IN int32 socket, OUT string buffer, IN int32 len)

설명: 현재 연결된 소캣(socket)에서 데이터를 특정 길이(len)만큼 읽어서 버퍼(buffer)에 저장한다. 만약, 함수가 호출된 시점에 지정된 크기(len)보 적은 양의 데이터만 수신된 상태라면 읽을 수 있을 만큼만 읽고 복귀 한다. 정해진 양을 모두 읽도록 하는 것은 NetReadTimeout 함수를 이용해야 함.

반환값: 읽은 문자열의 개수

(12) int32 NetWrite(IN int32 socket, IN string buffer, IN int32 len)

설명: 현재 연결된 소캣(socket)에서 데이터를 특정 길이 만큼 쓴다. 만약, 함수 호출 시점에 특정 크기(len)를 쓸 수 없는 상황이라면, 쓰기 가능한 데이터만큼만 쓰고, 복귀 한다. 전해진 양을 모두 전달 하려면 NetWriteTimeout 함수를 이용해야 함.

반환값: 씌여진 문자 개수

(13) int32 NetClose(IN int32 socket)

설명: 현재 연결된 접속을 해제 한다.

반환값: 성공하면 0, 실패하면 -1

(14) int32 NetTCPListenSocket(IN uint16 port, IN string bindaddr)

설명: 지정된 포트번호(port)로 TCP 포트를 바인딩 해서 반환 한다. 바인딩 주소는 bindaddr 에서 지정한다. 모든 주소에 대해 binding 할 경우, bindaddr은 INADDR_ANY 를 지정 한다.

반환값: 포트 바인딩에 성공하면 성공한 소캣 번호 반환, 실패하면 0이하의 값

(15) int32 socket(IN int32 domain, IN int32 type, IN int32 protocol)

설명: 소켓을 연다. domain = AF_INET, AF_INET6 등등.. type = SOCK_STREAM, SOCK_PACKET..

반환값: 소켓 번호 반환

(16) int32 bind(IN int32 socket, IN int32 domain, IN string bindaddr, IN uint16 port)

설명: 열린 소캣(socket)에 대해 주소와 포트를 바인딩 한다.

반환값: 바인딩 성공 여부. 성공하면 0 반환, 실패하면 0이 아닌 값 반환

(17) int32 listen(IN int32 socket, IN int32 backlog)

설명: 바인딩된 소캣(socket)에 대해 리스닝 상태를 설정 한다.

반환값: 성공하면 0, 실패하면 0이 아닌 값

(18) int32 accept(IN int32 socket, OUT string clientIP, OUT uint16 clientPort)

설명: 리스닝되어 있는 소켓에서 새로운 연결을 기다린다. clientIP와 clientPort에는 접속된 clinet의 IP주소와 Port번호가 저장된다

반환값: 새로 연결된 client의 소캣 번호

(19) void closesocket(IN int32 socket)

설명: 열려 있는 소캣을 닫는다.

(20) int32 NetReadTimeout(IN int32 socket, OUT string buffer, IN int32 len, IN int32 timeout)

설명: 연결된 소캣(socket)에서 데이터를 특정 길이(len)만큼 읽어서 buffer에 저장 한다. 만약, 호출된 시점에서 데이터가 len 만큼 수신 되지 못했다 해도, 최대 시간(초, timeout) 까지 기다린다.

만약, timeout 시간이 되도록 데이터가 수신되지 못하면 현재 읽은 부분까지만 반환 한다.

반환값: 읽어온 데이터 길이

(21) int32 NetWriteTimeout(IN int32 socket, IN string buffer, IN int32 len, IN int32 timeout)

설명: 연결된 소캣(socket)에 데이터를 쓴다. 함수가 호출된 시점에서 전체 데이터 길이(len)만큼 보낼 수 없다 하더라도, 최대 기다리는 시간(초, timeout)까지 대기 해서 len 만큼 데이터를 보낸다. timeout 시간이 넘어가면 그냥 복귀 한다.

반환값: 전손항 데이터 길이

(22) FD_ZERO(OUT handle fds)

설명: 여러 소캣들의 비트셑을 초기화 한다.

(23) FD_SET(IN int32 fd, OUT handle fds)

설명: 초기화된 fdset에 지정된 소캣(fd)을 추가한다.

(24) byte FD_ISSET(IN int32 fd, IN handle fds)

설명: 지정된 소캣(fd)의 이벤트가 발생했는지 검사한다.

반환값: 만약 이벤트가 발생했으면 TRUE 반환, 아니면 FALSE 반환

(25) FD_CLOSE(IN handle fds)

설명: 소켓 셑의 메모리를 해제한다.

(26) int32 select(IN int32 maxfd, IN handle rfds, IN handle wfds, IN handle efds, IN int32 timeout_sec, IN int32 timeout_usec)

설명: 지정된 시간(초, timeout_sec / 마이크로초, timeout_usec) 만큼 대기 하여, 소켓 이벤트가 발생하는지 확인 한다.

rfds 는 읽기 이벤트 fdset 핸들, wfds 는 쓰기 이벤트 fdset 핸들, efds는 에러 이벤트 fdset 핸들을 지정한다.

반환값: -1이 반환되면 에러 발생, 0이면 timeout이 지나도록 아무런 이벤트가 발생하지 않음. 양수가 반환되면 특정 소켓에서 이벤트가 발생함.

(27) handle THNet_FTPClient_GetHandle(void)

설명: ftp 클라이언트 핸들을 생성 한다

반환값: ftp 클라이언트 핸들

(28) void THNet_FTPClient_SetUserID(IN handle ftpHandle, IN string userid)

설명: ftp 클라이언트에 사용자 ID를 설정 한다

(29) void THNet_FTPClient_SetPassword(IN handle ftpHandle, IN string password)

설명: ftp 클라이언트에 사용자 비밀번호를 설정 한다.

(30) void THNet_FTPClient_SetServerAddr(IN handle ftpHandle, IN string serveraddr)

설명: ftp 클라이언트에 FTP서버 주소를 설정 한다.

(31) void THNet_FTPClient_SetServerPort(IN handle ftpHandle, IN uint16 serverport)

설명: ftp 클라이언트에 FTP서버 주소포트를 설정 한다.

(32) void THNet_FTPClient_SetTimeout(IN handle ftpHandle, IN int32 i32Timeout)

설명: ftp 클라이언트의 시간 제한을 설정 한다.

(33) void THNet_FTPClient_DoLogin(IN handle ftpHandle)

설명: ftp 클라이언트의 login을 실시 한다.

(34) int64 THNet_FTPClient_GetFileList(IN handle ftpHandle, IN string path, OUT string buffer)

설명: ftp 클라이언트 핸들을 이용하여 FTP서버의 path 경로에 있는 파일 목록 결과를 buffer에 저장 한다.

반환값: 버퍼에 저장된 데이터 크기

(35) int64 THNet_FTPClient_GetFilenameList(IN handle ftpHandle, IN string path, OUT string buffer)

설명: ftp 클라이언트 핸들을 이용하여 FTP서버의 path 경로에 있는 파일 목록만 buffer에 저장한다.

반환값: 버퍼에 저장된 데이터 크기

(36) int64 THNet_FTPClient_GetFileToBuffer(IN handle ftpHandle, IN string path, OUT string buffer)

설명: ftp 서버에 path 경로의 파일을 버퍼에 저장한다.

반환값: 받아온 파일의 크기

(37) int64 THNet_FTPClient_GetFile(IN handle ftpHandle, IN string path, IN string savefilename)

설명: ftp 서버의 path 경로의 파일을 savefilename으로 저장 한다.

반환값: 받아온 파일의 크기

(38) int64 THNet_FTPClient_GetFileSize(IN handle ftpHandle, IN string path)

설명: FTP서버의 path에 지정된 파일의 크기를 획득 한다.

반환값: 파일의 크기

(39) int64 THNet_FTPClient_PutFIleFromBuffer(IN handle ftpHandle, IN string path, IN string buffer)

설명: 버퍼(buffer)의 내용을 FTP서버에 지정된 path 경로에 파일을 업로드 한다.

반환값: 업로드된 파일 크기

(40) int64 THNet_FTPClient_PutFile(IN handle ftpHandle, IN string path, IN string filename)

설명: 지정된 filename을 FTP서버에 path 경로로 파일을 업로드 한다.

반환값: 업로드된 파일 크기

(41) void THNet_FTPClient_SetDir(IN handle ftpHandle, IN string path)

설명: FTP서버의 현재 작업 경로를 설정 한다

(42) string THNet_FTPClient_GetCurrentDir(IN handle ftpHandle)

설명: FTP서버에서 현재의 작업 디렉토리를 획득 한다.

반환값: 디렉토리 경로

(43) void THNet_FTPClient_CloseHandle(IN handle ftpHandle)

설명: FTP 서버의 접속을 해제 한다.

(44) handle THNet_HTTPClient_GetHandle(void)

설명: HTTP 클라이언트 핸들을 획득 한다.

반환값: HTTP 클라이언트 핸들

(45) byte THNet_HTTPClient_SetURL(IN handle httpHandle, IN string url)

설명: HTTP 클라이언트에 URL을 설정 한다.

반환값: 성공하면 TRUE, 실패하면 FLASE를 반환 한다.

(46) byte THNet_HTTPClient_SetProxyServer(IN handle httpHandle, IN string proxyaddr)

설명: ProxyServer의 주소를 설정 한다.

반환값: 성공하면 TRUE, 실패하면 FALSE를 반환 한다.

(47) byte THNet_HTTPClient_SetProxyPort(IN handle httpHandle, IN uint16 proxyport)

설명: ProxyServer의 포트를 설정 한다.

반환값: 성공하면 TRUE, 실패하면 FALSE를 반환 한다.

(48) void THNet_HTTPClient_SetTimeout(IN handle httpHandle, IN int32 i32Timeout)

설명: http 클라이언트의 시간 제한 값을 설정 한다.

(49) void THNet_HTTPClient_ResetBrowser(IN handle httpHandle)

설명: 웹 브라우저 클라이언트를 초기화 한다. (모든 데이터를 초기화 함)

(50) void THNet_HTTPClient_ResetPage(IN handle httpHandle)

설명: 새로운 페이지를 Open 하는 걸로 설정 한다. (Cookie 값이나, Referer는 변경되지 않음)

(51) int64 THNet_HTTPClient_GetContentsToBuffer(IN handle httpHandle, IN int32 method, OUT string buffer)

설명: 지정된 method 방식(TH_HTTP_METHOD_GET/ TH_HTTP_METHOD_POST)로 현재 설정된 URL의 내용을 받아와서 버퍼(buffer)에 저장 한다.

반환값: 받아온 버퍼의 크기

(52) int64 THNet_HTTPClient_GetContents(IN handle httpHandle, IN int32 method, IN string filename)

설명: 지정된 method 방식(TH_HTTP_METHOD_GET/ TH_HTTP_METHOD_POST)로 현재 설정된 URL의 내용을 받아와서 지정된 파일에 저장한다.

반환값: 받아온 파일의 크기

(53) void THNet_HTTPClient_RequestHeader_Set(IN handle httpHandle, IN string name, IN string value)

설명: HTTP Request 시 서버로 전달되는 해더를 설정 한다.

(54) string THNet_HTTPClient_RequestHeader_Get(IN handle httpHandle, IN string name)

설명: HTTP Request시 서버로 전달되는 해더의 값을 획득 한다.

반환값: 해더 값 문자 버퍼

(55) void THNet_HTTPClient_RequestHeader_SetDefault(IN handle httpHandle)

설명: HTTP Request의 요청 해더를 기본 값으로 저장 한다.

(56) string THNet_HTTPClient_ReplyHeader_Get(IN handle httpHandle, IN string name)

설명: HTTP 응답 해더 값을 획득 한다.

반환값: 지정된 name으로 설정된 응답 해더 값

(57) void THNet_HTTPClient_UserValue_Set(IN handle httpHandle, IN string name, IN string value)

설명: POST 혹은 GET 요청시 서버로 보내는 사용자 데이터를 설정 한다.

(58) string THNet_HTTPClient_UserValue_Get(IN handle httpHandle, IN string name)

설명: 사용자 데이터 값을 획득 한다.

(59) void THNet_HTTPClient_UserValue_Reset(IN handle httpHandle)

설명: 사용자 데이터 값을 모두 삭제 한다.

(60) void THNet_HTTPClient_Cookie_Set(IN handle httpHandle, IN string name, IN string value)

설명: HTTP 클라이언트에 쿠키 값을 설정 한다.

(61) string THNet_HTTPClient_Cookie_Get(IN handle httpHandle, IN string name)

설명: HTTP 클라이언트에서 쿠키 값을 획득 한다.

(62) void THNet_HTTPClient_Cookie_Reset(IN handle httpHandle)

설명: 크키 값을 모두 삭제 한다.

(63) void THNet_HTTPClient_Cookie_CloseHandle(IN handle httpHandle)

설명: HTTP 클라이언트를 삭제한다.

(64)

10.비트 처리 함수

(1) string BITPROC_MakeBitString(IN string str, IN int32 strlen, IN int32 nIntervalByte, IN uint8 LsbMsbType, IN uint8 bitCount)

설명: 문자열(str)에서 nIntervalByte 단위로 건너뛰어서 strlen길이 까지 가도록, bitCount 만큼 LSB 혹은 MSB 비트를 떼어내서 새로운 스트링 문자열로 만들어 반환 한다.

예) string dest = BITPROC_MakeBitString(str, strlen(str), 1, TH_BITPROC_LSB, 2)

--> str의 처음부터 끝까지 모든 바이트에서(1바이트 단위), 뒤쪽 2비트씩 복사해서 새로운 스트링 값을 반환 한다.

string dest = BITPROC_MakeBitString(str, strlen(str), 3, TH_BITPROC_MSB, 1)

--> str의 처음부터 끝까지 3 * n 번째 바이트마다 앞쪽 1비트씩 복사해서 새로운 스트링을 반환 한다.

반환값: 새로운 비트로 구성된 문자열

(2) string BITPROC_RightShiftString(IN string str, IN int32 index)

설명: 문자열(str)의 전체 비트를 오른쪽으로 index 만큼 쉬프트하여 새로운 문자열 변수로 반환 한다.

반환값: 새로운 비트로 구성된 문자열

(3) string BITPROC_LeftShiftString(IN string str, IN int32 index)

설명: 문자열(str)의 전체 비트를 왼쪽으로 index 만큼 쉬프트하여 새로운 문자열 변수로 반환 한다.

반환값: 새로운 비트로 구성된 문자열.

(4) string BITPROC_XorMask(IN string str, IN int32 strlen, IN string xorstr, IN int32 xorlen)

설명: 지정된 문자 버퍼(str) 처음부터 strlen 길이 만큼, xorstr에 있는 문자를 차례로 XOR 연산 시켜서 새로운 변수로 반환한다.

반환값: xor 연산이된 결과

(5)

11. 이미지 처리 함수

가. 이미지 함수 관련 상수 값 목록

- 이미지 타입

TH_IMAGE_TYPE_BMP

TH_IMAGE_TYPE_GIF

TH_IMAGE_TYPE_JPG

TH_IMAGE_TYPE_PNG

TH_IMAGE_TYPE_ICO

TH_IMAGE_TYPE_TIF

TH_IMAGE_TYPE_TGA

TH_IMAGE_TYPE_PCX

TH_IMAGE_TYPE_WBMP

TH_IMAGE_TYPE_WMF

TH_IMAGE_TYPE_JP2

TH_IMAGE_TYPE_JPC

TH_IMAGE_TYPE_PGX

TH_IMAGE_TYPE_PNM

TH_IMAGE_TYPE_RAS

TH_IMAGE_TYPE_JBG

TH_IMAGE_TYPE_MNG

TH_IMAGE_TYPE_SKA

TH_IMAGE_TYPE_RAW

TH_IMAGE_TYPE_PSD

- OVerflow 타입

TH_IMAGE_OVERFLOW_COLOR

TH_IMAGE_OVERFLOW_BACKGROUND

TH_IMAGE_OVERFLOW_TRANSPARENT

TH_IMAGE_OVERFLOW_WRAP

TH_IMAGE_OVERFLOW_REPEAT

TH_IMAGE_OVERFLOW_MIRROR

- Interpolation 타입

TH_IMAGE_INTERPOLATION_NEAREST

TH_IMAGE_INTERPOLATION_BILINEAR

TH_IMAGE_INTERPOLATION_BSPLINE

TH_IMAGE_INTERPOLATION_BICUBIC

TH_IMAGE_INTERPOLATION_BICUBIC2

TH_IMAGE_INTERPOLATION_LANCZOS

TH_IMAGE_INTERPOLATION_BOX

TH_IMAGE_INTERPOLATION_HERMITE

TH_IMAGE_INTERPOLATION_HAMMING

TH_IMAGE_INTERPOLATION_SINC

TH_IMAGE_INTERPOLATION_BLACKMAN

TH_IMAGE_INTERPOLATION_BESSEL

TH_IMAGE_INTERPOLATION_GAUSSIAN

TH_IMAGE_INTERPOLATION_QUADRATIC

TH_IMAGE_INTERPOLATION_MITCHELL

TH_IMAGE_INTERPOLATION_CATROM

TH_IMAGE_INTERPOLATION_HANNING

TH_IMAGE_INTERPOLATION_POWER

나. 함수들

(1) handle THImageTools_CreateFromFile(IN string filename)

설명: 지정된 파일명에서 이미지 핸들을 생성한다.

반환값: 이미지 handle 값

(2) handle THImageTools_Create(IN int32 width, IN int32 height, IN int32 bpp)

설명: 가로 세로 크기가 width, height인 bpp 비트 이미지를 생성한다. (빈 이미지)

반환값: 이미지 handle 값

(3) void THImageTools_Free(IN handle imgHandle)

설명: 이미지 핸들의 메모리를 해제 한다

반환값: 없음

(4) handle THImageTools_LoadFromRGB(IN string rgb)

설명: rgb 비트 데이터에서 이미지 핸들을 만든다.

반환값: 이미지 handle 값

(5) handle THImageTools_LoadFromRGBA(IN string rgba)

설명: rgba 비트 데이터에서 이미지 핸들을 만든다. (rgb + alpha 값)

반환값: 이미지 handle 값

(6) int32 THImageTools_GetType(IN handle iHandle)

설명: 이미지 타입을 반환 한다

반환값: 이미지 타입 값

(7) byte THImageTools_IsValid(IN handle iHandle)

설명: 이미지 핸들이 올바른지 확인 한다.

반환값: 이미지 핸들이 정상적이면 TRUE, 에러가 있으면 FALSE 반환

(8) void THImageTools_ConvertThumnail(IN handle iHandle, IN int32 width, IN int32 height)

설명: 이미지 핸들(iHandle)의 이미지를 지정된 가로(width), 세로(height) 크기의 썸네일로 만든다

(9) void THImageTools_Save(IN handle iHandle, IN string filename, IN int32 type)

설명: 이미지 핸들(iHandle)로 지정된 이미지 데이터를 지정된 타입(type)의 이미지 형식으로 저장한다.

(10) void THImageTools_GetImageData(IN handle iHandle, OUT string buffer, OUT int32 size)

설명: 이미지에서 RGBA 데이터를 획득 한다.

(11) int32 THImageTools_GetImageHeight(IN handle iHandle)

설명: 이미지의 세로 크기를 획득한다.

반환값: 세로 픽셀 개수

(12) int32 THImageTools_GetImageWidth(IN handle iHandle)

설명: 이미지의 가로 크기를 획득 한다.

반환값: 가로 픽셀 개수

(13) int32 THImageTools_GetXDPI(IN handle iHandle)

설명: 이미지의 가로 DPI 값을 획득 한다.

반환값: 가로의 DPI 값

(14) int32 THImageTools_GetYDPI(IN handle iHandle)

설명: 이미지의 세로 DPI 값을 획득 한다.

반환값: 세로의 DPI 값

(15) void THImageTools_SetXDPI(IN handle iHandle, IN int32 dpi)

설명: 이미지의 가로 DPI 값을 설정 한다.

(16) void THImageTools_SetYDPI(IN handle iHandle, IN int32 dpi)

설명: 이미지의 세로 DPI 값을 설정 한다.

(17) int32 THImageTools_GetBPP(IN handle iHandle)

설명: 이미지의 BPP 값을 획득 한다.

반환값: BPP 값

(18) int32 THImageTools_GetJpgQuality(IN handle iHandle)

설명: 이미지가 JPG일 경우, JPQ 퀄리티 값을 획득 한다

반환값: 이미지 퀄리티 값

(19) void THImageTools_SetJpgQuality(IN handle iHandle)

설명: 이미지를 JPG일 경우, JPG 퀄리티 값을 설정한다.

(20) void THImageTools_Resize(IN handle iHandle, IN int32 width, IN int32 height, IN int32 interpolation, IN int32 overflow)

설명: 이미지의 크기를 조정한다. 이때 크기 변화에 따른 이미지 픽셀은 interpolation, overflow 방식을 지정한다

(21) void THImageTools_SplitRGB(IN handle iHandle, OUT handle rHandle, OUT handle gHandle, OUT handle bHandle)

설명: 이미지를 r,g,b 색 분류로 나누어 각각의 이미지 핸들에 저장한다

(22) void THImageTools_SplitCMYK(IN handle iHandle, OUT handle cHandle, OUT handle mHandle, OUT handle yHandle, OUT handle kHandle)

설명: 이미지를 c,m,y,k 색 분류로 나누어 각각의 이미지 핸들에 저장한다.

(23) void THImageTools_SplitHSL(IN handle iHandle, OUT handle hHandle, OUT handle sHandle, OUT handle lHandle)

설명: 이미지를 h,s,l 색 분류로 나누어 각각의 이미지 핸들에 저장한다.

(24) void THImageTools_SplitXYZ(IN handle iHandle, OUT handle xHandle, OUT handle yHandle, OUT handle zHandle)

설명: 이미지를 x,y,z 색 분류로 나누어 각각의 이미지 핸들에 저장한다.

(25) void THImageTools_SplitYIQ(IN handle iHandle, OUT handle yHandle, OUT handle iHandle, OUT handle qHandle)

설명: 이미지를 y,i,q 색 분류로 나누어 각각의 이미지 핸들에 저장한다.

(26) void THImageTools_SplitYUV(IN handle iHandle, OUT handle yHandle, OUT handle uHandle, OUT handle vHandle)

설명: 이미지를 y,u,v 색 분류로 나누어 각각의 이미지 핸들에 저장한다.

(27) void THImageTools_SetGrayScale(IN handle iHandle)

설명: 이미지를 그레이스케일로 변경 한다.

(28) void THImageTools_SetMirror(IN handle iHandle)

설명: 이미지의 좌우를 뒤집는다.

(29) void THImageTools_SetFlip(IN handle iHandle)

설명: 이미지의 상하를 뒤집는다.

(30) void THImageTools_SetNegative(IN handle iHandle)

설명: 이미지의 음영을 바꾼다.

(31) void THImageTools_TransEnhancedRGB(IN handle iHandle, IN int32 rgbopt, IN int32 bLSB, IN int32 setV)

설명: 이미지의 RGB 값의 LSB, MSB 값에 따라서 해당 R,G,B 값을 설정 한다.

rgbopt 는 TH_IMAGE_TRANS_ENHANCED_RGB_R, TH_IMAGE_TRANS_ENHANCED_RGB_G, TH_IMAGE_TRANS_ENHANCED_RGB_B 를 조합해서 사용할 수 있으며, bLSB가 TRUE 면, Last signification bit 값이 1일 경우, FALSE면, Most signification bit 값이 1일 경우에 해당 RGB 값을 setV로 변경 한다.

(32) int32 THImageTools_GetEnhancedRGBAvg(IN handle iHandle, IN int32 rgbopt, IN int32 dir, IN int32 pos)

설명: 이미지의 RGB 값의 평균을 구한다. rgbopt는 THImageTools_TransEnhancedRGB() 함수에서 사용하는 값과 동일 하며, dir 에는 TH_IMAGE_TRANS_HORZ일 경우 가로 픽셀에 대한 평균 값을, TH_IMAGE_TRANS_VERT 일 경우 세로 픽셀에 대한 평균 값을 구한다. 이때 기준이 되는 위치는 pos 값. 만약, dir 값이 TH_IMAGE_TRANS_ALL 일 경우에는 전체 이미지에 대한 평균 값을 구한다.

(33) void THImageTools_TransPinch(IN handle iHandle, IN int32 factor)

설명: 이미지를 pinch 변형 시킨다.

(34) void THImageTools_TransPunch(IN handle iHandle, IN int32 factor)

설명: 이미지를 punch 변형 시킨다.

(35) void THImageTools_TransTwirl(IN handle iHandle, IN int32 factor)

설명: 이미지를 twirl 변형 시킨다.

(36) void THImageTools_TransCylinder(IN handle iHandle, IN int32 factor)

설명: 이미지를 cylinder 변형 시킨다.

(37) void THImageTools_TransBathroom(IN handle iHandle, IN int32 factor)

설명: 이미지를 화장실 효과(?) 변형 시킨다

(38) void THImageTools_TransSkew(IN handle iHandle, IN float32 x, IN float32 y, IN int32 xp, IN int32 yp, IN byte bInterpolation)

설명: 이미지를 기울이는 변형을 한다

(39) void THImageTools_TransRotate(IN handle iHandle, IN float32 angle, IN int32 interpolation_method, IN int32 overflow_method, IN bkcolor, iIN byte bOptAngle, IN byte bOptSize)

설명: 이미지를 회전 시킨다

(40) void THImageTools_TransRotateSimple(IN handle iHandle, IN float32 angle)

설명: 이미지를 단순 회전 시킨다

(41) void THImageTools_Trans3DRotate(IN handle iHandle, IN float32 x, IN float32 y, IN float32 z, IN int32 type)

설명: 이미지를 3D 회전 한다.

(42) void THImageTools_TransCrop(IN handle iHandle)

설명: 현재 선택되어 있는 선택 영역을 남기고, 나머지를 잘라내는 Crop 변형 시킨다.

(43) void THImageTools_TransWave(IN handle iHandle, IN int32 swing, IN int32 freq)

설명: 이미지를 Wave 변형 시킨다.

(44) void THImageTools_EffectLight(IN handle iHandle, IN int32 factor)

설명: factor 는 -255 에서 255 사이 값

(45) void THImageTools_EffectContrast(IN handle iHandle, IN int32 factor)

설명: factor는 -100 에서 100 사이 값

(46) void THImageTools_EffectGamma(IN handle iHandle, IN float32 factor)

(47) void THImageTools_EffectColorizeHSL(IN handle iHandle, IN uint8 hvalue, IN uint8 svalue, IN float32 blend)

(48) void THImageTools_EffectShiftRGB(IN handle iHandle, IN int32 r, IN int32 g, IN int32 b)

(49) void THImageTools_EffectThreshold(IN handle iHandle, IN uint8 factor)

(50) void THImageTools_EffectRedEyeRemove(IN handle iHandle)

(51) void THImageTools_EffectNoise(IN handle iHandle, IN int32 level)

(52) void THImageTools_EffectMosaic(IN handle iHandle, IN int32 blocksize)

(53) void THImageTools_EffectGlassTile(IN handle iHandle, IN int32 xtile, IN int32 ytile)

(54) void THImageTools_EffectNoisify(IN handle iHandle, IN int32 level)

(55) void THImageTools_EffectEmboss(IN handle iHandle, IN int32 level)

(56) void THImageTools_EffectSolarize(IN handle iHandle, IN int32 level)

(57) void THImageTools_EffectPosterize(IN handle iHandle, IN int32 level)

(58) void THImageTools_EffectOilpaint(IN handle iHandle, IN int32 level)

(59) void THImageTools_Effect3DGrids(IN handle iHandle, IN int32 size, IN int32 depth)

(60) void THImageTools_EffectImpressLight(IN handle iHandle, IN int32 level)

(61) void THImageTools_EffectImpressDark(IN handle iHandle, IN int32 level)

(62) void THImageTools_EffectUnsharpMask(IN handle iHandle, IN float32 radius, IN float32 amount, IN int32 threshold)

(63) void THImageTools_EffectIllusion(IN handle iHandle, IN int32 level)

(64) void THImageTools_BPPIncrease(IN handle iHandle)

설명: 이미지의 BPP를 한단계 높인다. 1 bit -> 8 bit -> 16 bit -> 24 bit -> 32 bit

(65) void THImageTools_BPPDecrease(IN handle iHandle)

설명: 이미지의 BPP를 한단계 내린다. 32 bit -> 24 bit -> 16 bit -> 8 bit -> 1bit

(66) void THImageTools_SetTransColor(IN handle iHandle, IN uint32 color)

설명: 이미지에서 투명색을 지정한다.

(67) byte THImageTools_IsInsidePixel(IN handle iHandle, IN int32 x, IN int32 y)

설명: 지정된 x, y 좌표가 이미지 안쪽에 있는지 검사한다.

(68) uint8 THImageTools_GetPixelIndex(IN handle iHandle, IN int32 x, IN int32 y)

설명: 지정된 x, y 좌표의 인덱스 픽셀을 획득 한다.

(69) uint32 THImageTools_GetColor(IN handle iHandle, IN int32 x, IN int32 y)

설명: 지정된 x, y 좌표의 rgb 색상을 획득 한다.

(70) void THImageTools_SelectionClear(IN handle iHandle)

설명: 지정된 선택 영역을 모두 지운다

(71) void THImageTools_SelectionAddRect(IN handle iHandle, IN int32 left, IN int32 top, IN int32 right, IN int32 bottom)

설명: 사각형 선택 영역을 추가 한다.

(72) handle THImageTools_SelectionCopyImage(IN handle iHandle, IN uint32 bkcolor)

설명: 선택 영역을 복사해서 이미지 핸들로 반환 한다.

(73) void THImageTools_SelectionCutImage(IN handle iHandle, IN uint32 bkcolor)

설명: 선택 영역을 잘라낸다. 잘라낸 영역은 bkcolor로 채운다

(74) void THImageTools_SelectionPasteImage(IN handle iHandle, IN handle iSrcHandle, IN byte bUseBkColor, IN uint32 bkcolor, IN byte bResize)

설명: 선택 영역에 지정된 이미지(iSrcHandle)을 복사한다. bResize가 TRUE 라면, 선택 영역 크기에 맞추어 크기 변화를 한 후 복사 하고, FALSE 라면, 선택 영역의 남은 공간은 bkcolor 를 채운다. (bUseBkColor가 TRUE일 경우).

(75) string THImageTools_GetBits(IN handle iHandle)

설명: 이미지의 RGB 비트를 획득 한다.

반환값: rgb bit 데이터

(76) uint32 THImageTools_GetBitsSize(IN handle iHandle)

설명: 이미지의 RGB 비트 데이터 크기를 반환 받는다.

반환값: rgb bit 데이터 크기(bytes 크기)

(77) uint32 RGB(IN uint8 r, IN uint8 g, IN uint8 b)

설명: R,G,B 색상 값을 이용하여 32비트 RGB 값을 만든다.

반환값: RGB 색상 값

(78) uint32 RGBA(IN uint8 r, IN uint8 g, IN uint8 b, IN uint8 a)

설명: R,G,B와 투명도값 A를 포함하여 32비트 RGB 값을 만든다.

반환값: RGB 색상 값

(79) uint8 RGBA_R(IN uint32 rgba)

설명: RGB값에서 R 값을 획득 한다.

반환값: R 값

(80) uint8 RGBA_G(IN uint32 rgba)

설명: RGB값에서 G 값을 획득 한다.

반환값: G값

(81) uint8 RGBA_B(IN uint32 rgba)

설명: RGB값에서 B 값을 획득 한다.

반환값: B값

(82) uint8 RGBA_A(IN uint32 rgba)

설명: RGB값에서 A 값을 획득 한다.

반환값: A 값

(83)

12.암호화 처리 함수

(1) handle Crypt_Init(IN int32 alg, IN string key, IN int32 keylen, IN int32 blockSize, IN string iv)

설명: 암호처리 루틴을 초기화 한다. 지정된 알고리즘(alg)을 지정된 블럭 크기(blockSize)로 초기화 한다.

iv 는 Initial vector, 암호화 알고리즘의 경우 암호 키(key)와 키 길이(keylen)를 지정해야 함.

■ alg 항목에 사용가능한 암호화 알고리즘

alg 값	알고리즘	가능한 블럭 크기(Bytes)	키 길이(Bytes)
TH_CRYPT_ALG_DES	DES 암호화 알고리즘	8	7
TH_CRYPT_ALG_RIJNDAEL	Rijndael 암호화 알고리즘	16	16
TH_CRYPT_ALG_RC4	RC4 암호화 알고리즘	16	16

TH_CRYPT_ALG_AES	AES 암호화 알고리즘	16 / 24 / 32	16 / 24 / 32
TH_CRYPT_ALG_CAST5	CAST5 암호화 알고리즘	8	16
TH_CRYPT_ALG_TWOFISH	Twofish 암호화 알고리즘	16	16
TH_CRYPT_ALG_BLOWFISH	Blowfish 암호화 알고리즘	8	8 ~ 56
TH_CRYPT_ALG_SERPENT	Serpent 암호화 알고리즘	16	32

■ alg 항목에 사용가능한 해쉬 알고리즘

alg값	알고리즘	블럭 크기(Bytes)	출력 크기(Bytes)
TH_CRYPT_ALG_MD5	MD5 해쉬 알고리즘	-	16
TH_CRYPT_ALG_SHA1	SHA1 해쉬 알고리즘	-	20
TH_CRYPT_ALG_SHA2	SHA2 해쉬 알고리즘	28(224bit) 32(256bit) 48(384bit) 64(512bit)	28 32 48 64
TH_CRYPT_ALG_NTLM	NTLM 해쉬 알고리즘	-	16
TH_CRYPT_ALG_WHIRLPOOL	WhirlPool 해쉬 알고리즘	-	64
TH_CRYPT_ALG_RMD160	RMD160 해쉬 알고리즘	-	20
TH_CRYPT_ALG_MYSQL3	MySQL3 패스워드 해쉬	-	8

■ 기타 알고리즘

TH_CRYPT_ALG_BASE64 BASE64 인코딩

반환값: 암호화 처리 핸들

(2) byte Crypt_DoCipher(IN handle cHandle, IN int32 mode, IN byte bEnc, IN string inStr, IN int32 inStrLen, OUT string outStr, OUT int32 outStrLen)

설명: 암호 루틴을 실행 한다. 지정된 암호 처리 루틴(cHandle)을 이용하여 mode 방식의 암호화 루틴을 실행 한다. bEnc가

TRUE면 암호화, FALSE면 복호화를 실행 한다. inStr은 입력 문자, inStrLen은 입력 문자의 길이, outStr은 결과 저장 버퍼, outStrLen은 결과 버퍼의 길이

■ Mode에서 사용할 수 있는 값은 다음과 같음

TH_CRYPT_MODE_NONE : 암호 알고리즘을 선택한것이 아닐 경우.

TH_CRYPT_MODE_ECB : ECB 모드 설정

TH_CRYPT_MODE_CFB : CFB 모드 설정

TH_CRYPT_MODE_CBC : CBC 모드 설정

TH_CRYPT_MODE_LWR : LRW 모드 설정

TH_CRYPT_MODE_XTS : XTS 모드 설정

반환값: 성공하면 TRUE, 실패하면 FALSE 반환

(3) void Crypt_Reset(IN handle cHandle)

설명: 암호 처리 핸들을 리셋 시킨다.

(4) void Crypt_Free(IN handle cHandle)

설명: 암호 처리 핸들을 삭제 한다.

(5) handle Crypt_AlgGetHandle(IN int32 alg)

설명: 알고리즘 핸들을 획득 한다. 개별 알고리즘을 핸들을 가져와서 조작하기 위한 함수. alg 값은 Crypt_Init() 에서 사용하

는 값과 동일 함.

반환값: 알고리즘 핸들값

(6) void Crypt_AlgFree(IN handle alg)

설명: 알고리즘 핸들을 해제 한다.

반환값: 없음

(7) byte Crypt_AlgSetKey(IN handle aHandle, IN string key, IN int32 keyLen)

설명: 알고리즘 핸들(aHandle)에 암호 키를 설정한다.

반환값: 성공하면 TRUE, 실패하면 FALSE

(8) byte Crypt_AlgSetInput(IN handle aHandle, IN string inBuffer, IN int32 bufferLen)

설명: 알고리즘 핸들(aHandle)에 입력 값을 설정 한다 만약, 해쉬 알고리즘 핸들이라면 해쉬 상태를 업데이트 한다.

반환값:성공하면 TRUE, 실패하면 FALSE

(9) byte Crypt_AlgDoIt(IN handle aHandle, OUT string buffer)

설명: 알고리즘 핸들(aHandle)로 지정된 알고리즘을 실행 하여, buffer에 결과를 저장 한다.

반환값: 성공하면 TRUE, 실패하면 FALSE

(10) void Crypt_AlgSetOption(IN handle aHandle, IN uint32 option, IN value)

설명: 알고리즘 핸들(aHandle)에 옵션 값을 설정 한다.

옵션(option)과 값(value)에 설정 할 수 있는 내용은 다음과 같다.

옵션	값	설명
TH_CRYPT_OPT_MODE	TH_CRYPT_MODE_NONE TH_CRYPT_MODE_ECB TH_CRYPT_MODE_CFB TH_CRYPT_MODE_CBC TH_CRYPT_MODE_LRW TH_CRYPT_MODE_XTS	암호 연산 방법 설정
TH_CRYPT_OPT_BLOCKSIZE	숫자	알고리즘에서 사용할 블럭 크기(Bytes)
TH_CRYPT_OPT_IV	Initial vector 설정	블럭 크기와 동일한 길이로 설정

(11) int32 Crypt_AlgGetOption(IN handle aHandle, IN uint32 option)

설명: 알고리즘의 옵션 값을 읽는다.

이 함수에서 사용할 수 있는 옵션 값은 다음과 같다

옵션	반환 값	설명
TH_CRYPT_OPT_MODE	TH_CRYPT_MODE_NONE TH_CRYPT_MODE_ECB TH_CRYPT_MODE_CFB TH_CRYPT_MODE_CBC TH_CRYPT_MODE_LRW TH_CRYPT_MODE_XTS	암호 연산 방법 설정
TH_CRYPT_OPT_KEYSIZE	숫자	설정된 키 크기
TH_CRYPT_OPT_INPUTSIZE	숫자	현재 입력된 길이

TH_CRYPT_OPT_OUTPUTSIZE	숫자	출력 길이

(12) void Crypt_AlgSetEncryptMode(IN handle aHandle, IN byte bEnc)

설명: bEnc값이 TRUE면, 암호화 작동 모드, FALSE면 복호화 작동 모드

(13)

13.플러그인 함수

(1) handle LoadLibrary(string path)

설명: DLL 혹은 Share Object(SO) 파일 라이브러리를 플러그인으로 읽는다.

반환값: 플러그인 핸들

(2) void FreeLibrary(IN handle pHandle)

설명: 읽어온 플러그인을 해제 한다.

(3)

14.메모리 관련 함수

(1) void memset(OUT value, IN char ch, IN int32 len, IN int32 offset)

설명: 지정된 변수(value)의 오프셋(offset)에서 지정된 길이(len)만큼 특정 문자(ch)로 채운다.

(2) void memcpy(OUT dest, IN source, IN int32 len, IN int32 dest0ffset, IN int32 src0ffset)

설명: 원본 변수(source)의 시작점(src0ffset)에서 부터 지정된 길이(len)만큼 복사해서 대상 변수(dest)의 시작점(dest0ffset)에 복사 한다.

(3) int32 memcmp(IN value1, IN value2, IN int32 len, IN int32 value1offset, IN int32 value2offset)

설명: 입력된 두개의 변수(value1, value2)의 각각의 시작점(value1offset, value2offset)에서 지정된 길이(len)만큼 같은지, 다른지 비교한다.

반환값: 같으면 0을 반환하고, 다르면 0이 아닌 값을 반환 한다.

(4) string malloc(IN int32 size)

설명: 메모리를 할당하여 스트링 변수로 반환 한다.

반환값: 할당된 메모리 영역을 가지는 스트링 변수

(5) int32 sizeof(IN value)

설명: 지정된 변수가 실제 메모리에서 차지하는 크기를 획득 한다.

반환값: 메모리 크기

(6) string mempack(IN value1, IN value2, ...)

설명: 입력으로 나열된 변수들(value1, value2,..)등을 차례로 변수의 실제 메모리 크기만큼 스트링 변수에 복사해서 하나의 스트링 변수로 만들어 반환 한다.

반환값: 나열된 변수들의 메모리 복사본을 가지는 스트링 변수

(7) int32 memunpack(IN packedStr, OUT value1, OUT value2, ...)

설명: 입력된 스트링(packedStr)에서 나열된 출력 변수(value1, value2...)등의 값을 획득 한다.

반환값: 획득된 변수의 개수

(8)

15.프로세스 관련 함수

(1) int32 system(IN string command)

설명: 시스템 명령어(command)를 실행 한다.

반환값: 시스템 명령어의 종료 코드

(2) handle popen(IN string command, IN string access)

설명: 시스템 명령어(command)를 파이프 출력이나 입력으로 변환 하여 실행 시킨다. access는 “r”, “w” 값을 지정한다.

반환값: 시스템 명령어 파이프 핸들

(3) int32 pread(IN handle pHandle, OUT string buffer, IN int32 len)

설명: 일기 전용(“r”)으로 열린 파이프 핸들에서 데이터를 읽는다.

반환값: 읽은 데이터 크기

(4) int32 pwrite(IN handle pHandle, IN string buffer, IN int32 len)

설명: 쓰기 전용(“w”)으로 열린 파이프 핸들에 데이터를 쓴다.

반환값: 씌여진 데이터 크기

(5) int32 peof(IN handle pHandle)

설명: 파이프가 닫혔는지 검사한다.

반환값: 닫혔다면, TRUE를 반환, 아니면 FALSE를 반환

(6) int32 pclose(IN handle pHandle)

설명: 파이프를 닫는다.

반환값: 에러발생하면 -1 반환, 성공하면 0 반환

(7) void sleep(IN int32 seconds)

설명: 지정된 초(seconds)만큼 프로세스를 유휴 상태로 만든다.

(8) void usleep(IN int32 uSeconds)

설명: 지정된 마이크로초(uSeconds)만큼 프로세스를 유휴 상태로 만든다.

(9)int32 fork(void);

설명: 자식 프로세스를 생성한다.

반환값: 자식 프로세스가 생성되면, 부모 프로세스에서는 자식 프로세스의 프로세스ID를 반환 받고, 자식 프록세스는 0을 반환 받는다.

(10) handle thread_create(func, arg1, arg2...)

설명: 새로운 쓰레드를 생성 합니다. func로 지정된 함수를 새로운 쓰레드로 생성하여 실행 시킵니다. arg1,arg2, 등의 전달인자는 func 의 함수 전달 인자에 할당되는 목록 입니다.

반환값: 쓰레드에 대한 핸들 값

(11) void thread_delete(handle tHandle)

설명: 모든 프로세스가 종료된 쓰레드의 핸들을 반환 합니다. tHandle로 명명된 쓰레드가 종료 되지 않은 상태에서 반환 할 경우, 에러가 발생 할 수 있습니다.

반환값: 없음

(12) int32 thread_isRunning(handle tHandle)

설명: tHandle로 지정된 쓰레드가 현재 실행 중인지 검사합니다.

반환값: 쓰레드가 실행중이면 TRUE, 종료되었으면 FALSE를 반환 합니다.

(13) Int32 native_call(IN string functionName, IN args...)

설명: 네이티브 인터페이스가 제공하는 함수를 호출 한다.. args 리스트는 최대 30개까지 전달 가능하다.

반환값: 네이트브 인터페이스 함수 호출 반환 값.

(14)

16.패턴 관련 함수

(1) handle THPattern_RegCompile(IN string regexp, IN int32 option, IN int32 maxmatched)

설명: 정규식 패턴을 컴파일 한다. maxmatched는 최대 매칭 개수를 지정. option은 pcre_compile() 함수와 동일하다.

반환값: 정규식 핸들

(2) int32 THPattern_IsRegMatched(IN handle pHandle, IN string data, IN int32 datalen, IN int32 StartOffset)

설명: data의 내용이 지정된 패턴에 맞는지 검사한다. datalen은 입력한 data의 길이, StartOffset은 검색을 시작할 data의 오프셋

반환값: 패턴에 맞는 스트링 개수

(3) int32 THPattern_RegGetSubString(IN handle pHandle, IN string data, IN int32 index, OUT string buffer)

설명: data에서 패턴에 해당하는 스트링을 획득 한다. index는 매칭 인덱스

반환값: buffer에 저장되는 스트링 길이

(4) void THPattern_RegClose(IN handle pHandle)

설명: 패턴 핸들을 닫는다.

(5)

17.날짜 관련 함수

(1) string THTimestamp_ConvertToString(IN uint64 timestamp, IN string formatstr)

설명: CCYYMMDDhhmmss 형식의 날짜 시간 스템프 숫자를 formatstr 에 맞추어 문자열로 반환 한다.

formatstr은 다음과 같은 스트링을 사용한다.

문자열	치환되는 값
YY	연도
MM	월
DD	일
hh	시각
mm	분
ss	초

반환값: 치환된 문자열 스트링

(2) int32 THTimestamp_DateWeek(IN int32 year, IN int32 month, IN int32 day)

설명: 지정된 년(year),월(month),일(day)의 요일을 획득 한다.

반환값: 0 - 6 까지의 값 (일요일 - 토요일)

(3) uint64 THTimestamp_MFTTimeToTimestamp(IN uint64 mfttime)

설명: MFT 시간 값(윈도우 64비트 시간값)을 CCYYMMDDhhmmss 형식의 타임 스템프로 변환 시킨다

반환값: 타임스템프 값

(4) uint64 THTimestamp_UnixTimeToTImestamp(IN uint32 unixtime)

설명: 유닉스 시간 값(32비트 time() 반환값)을 CCYYMMDDhhmmss 형식의 타임 스템프로 변환 시킨다.

반환값: 타임스템프 값

(5) uint64 THTimestamp_MacTimeToTimestamp(IN uint32 mactime)

설명: 맥 시간 값(32비트)을 CCYYMMDDhhmmss 형식의 타임 스템프로 변환 한다.

반환값: 타임스템프 값

(6) int32 THTimestamp_GetTotalDaysCountOfYear(IN int32 year, IN int32 month, IN int32 day)

설명: 지정된 연도(year)의 1월 1일 부터 지정된 날짜(month / day)까지의 날짜 수를 반환 한다.

반환값: 지정된 연도의 날짜까지의 날 수

(7) int64 THTimestamp_GetTotalDaysCountBetweenOfDate(IN int32 syear, IN int32 smon, IN int32 sday, IN int32 eyear, IN int32 emon, IN int32 eday)

설명: 시작 날짜(syear / smon / sday)부터 끝 날짜(eyear / emon / eday)까지의 날짜를 계산 해서 반환 한다.

반환값: 시작 날짜부터 끝 날짜까지의 날 수

(8) int64 THTimestamp_GetTotalMinuteBetweenOfTimestamp(IN int32 syear, IN int32 smon, IN int32 sday, IN int32 shour, IN int32 smin, IN int32 eyear, IN int32 emon, IN int32 eday, IN int32 ehour, IN int32 emin)

설명: 시작 날짜(syear / smon / sday) 와 시작 시각(shour:smin) 부터 끝 날짜(eyear / emon / eday), 끝 시각(ehour:emin)까지의 시간을 분(Minutes)으로 계산해서 반환 한다.

반환값: 시작 시각과 끝 시각 사이의 분(Minutes) 으로 계산한 숫자

(9) uint64 lib_THTimestamp_GetTimestampFromTimestampByMinute(IN int64 min, IN int32 year, IN int32 month, IN int32 day, IN int32 hour, IN int32 min)

설명: 지정된 날짜(year / month / day)의 시각(hour:min)에서부터 지정된 분(min)의 시간이 지난 후의 날짜를 타임 스템프 값으로 반환 한다.

반환값: 타임스템프값

(10) int32 THTimestamp_GetGanjiFromDate(IN int32 year, IN int32 month, IN int32 day)

설명: 지정된 날짜의 간지를 획득 한다.

반환값: 0-59 사이의 값. 간지 스트링 (g_strGanji) 배열값

(11) void THTimestamp_GetLunarDate(IN int32 year, IN int32 month, IN int32 day, OUT uint64 thisHapDate, OUT uint64 thisMangDate, OUT uint64 nextHapDate)

설명: 지정된 양력 날짜(year/month/day) 기준으로 이번달 합삭과 망, 다음달 합삭 날짜를 타임스템프 값으로 획득 한다.

(12) void THTimestamp_GetGanjiTimestamp(IN int32 year, IN int32 month, IN int32 day, IN int32 hour, IN int32 min, OUT int32 gyear, OUT int32 gmonth, OUT int32 gday, OUT int32 ghour)

설명: 지정된 날짜의 년/월/일/시에 대한 간지 값 인덱스를 획득 한다.(0-59 사이의 값)

(13) void THTimestamp_Get24Datestamp(IN int32 year, IN int32 month, IN int32 day, OUT int32 thisJpName, OUT uint64 thisJpDate, OUT int32 thisMdName, OUT uint64 thisMdDate, OUT uint64 nextJpName, OUT uint64 nextJpDate)

설명: 지정된 날짜의 24절기 이름과 날짜에 대한 스템프를 획득 한다.

출력 변수	설명
thisJpName	이번달 절입 이름(24절기 이름)
thisJpDate	이번달 절입 날짜에 대한 스템프
thisMdName	이번달 중기 이름(24절기 이름)
thisMdDate	이번달 중기 날짜에 대한 스템프
nextJpName	다음달 절입 이름(24절기 이름)
nextJpDate	다음달 절입 날짜에 대한 스템프

(14) void THTimestamp_SolorToLunar(IN int32 year, IN int32 month, IN int32 day, OUT int32 lyear, OUT int32 lmonth, OUT

int32 lday, OUT byte yundal, OUT byte large)

설명: 양력 날짜 (year/month/day)에 대해 음력 날짜(lyear/lmonth/lday)로 변환 한다.

yundal : 윤달일 경우 TRUE, 아니면 FALSE가 저장됨

large: 큰달일 경우 TRUE, 평달일 경우 FALSE가 저장됨

(15) void THTimestamp_LunarToSolor(IN int32 lyear, IN int32 lmonth, IN int32 lday, IN byte yundal, OUT int32 year, OUT int32 month, OUT int32 day)

설명: 음략 날짜(lyear/lmonth/lday)에 대해 양력 날짜(year/month/day)로 변환 한다. 만약, 음력이 윤달일 경우 yundal 값에 TRUE를 설정 한다.

(16)

18.아카이브 관련 함수

아카이브 라이브러리는 압축 파일을 입출력 하는 기능을 담당합니다. 현재는 지원되는 아카이브가 zip, tar 입니다. 아카이브 관련 핸들은 총 3가지 종류가 있습니다.

ArchiveManager_Create()로 획득되는 아카이브 파일 입출력 담당 아카이브 핸들, ArchiveManager_GetFirstEntry() 등으로 획득하는 아카이브 파일 내의 엔트리 핸들, ArchiveFile_Open() 등으로 획득하는 아카이브내의 파일 입출력을 담당하는 아카이브 파일 핸들이 있습니다. 핸들을 혼돈하지 않게 주의를 요합니다.

(1) handle ArchiveManager_Create(void)

설명: 아카이브 핸들을 생성 합니다.

반환값: 아카이브 핸들값

(2) void ArchiveManager_SetPassword(IN handle aHandle, IN string password)

설명: 아카이브 파일(zip)에 암호가 걸려 있을 경우, 패스워드(password)를 설정 합니다. 첫번째 전달인자는 아카이브 핸들 입니다. 패스워드 설정은 아카이브 파일을 Open 하기 전에 미리 설정되어 있어야 합니다.

(3) int32 ArchiveManager_Open(IN handle aHandle, IN string filename, IN int32 option)

설명: 아카이브 파일을 Open 합니다. option 값은 다음과 같습니다. 다음과 같은 옵션이 사용 가능 합니다.

TH_ARCHIVE_MANAGER_OPEN_READ	읽기 모드로 아카이브를 Open 합니다.
TH_ARCHIVE_MANAGER_OPEN_CREATE	아카이브 파일을 생성하는 옵션
TH_ARCHIVE_MANAGER_OPEN_APPEND	아카이브 파일에 새로운 파일을 추가 합니다.

- 현재는 아카이브 파일의 읽기 모드만 지원 합니다.

반환값: 성공하면 TRUE, 실패하면 FALSE 를 반환 합니다.

(4) void ArchiveManager_Close(IN handle aHandle)

설명: 아카이브 파일을 닫는다.

(5) handle ArchiveManager_GetFirstEntry(IN handle aHandle)

설명: 아카이브에 있는 첫번째 엔트리를 획득 한다.

반환값: 엔트리 핸들. (아카이브 핸들과 다른 핸들임)

(6) handle ArchiveManager_GetNextEntry(IN handle aHandle)

설명: 다음 엔트리 핸들을 획득 한다.

반환값: 엔트리 핸들

(7) handle ArchiveManager_GetEntry(IN handle aHandle, IN string filename)

설명: 입력된 filename 경로에 있는 엔트리 핸들을 획득 한다.

반환값: 엔트리 핸들

(8) void ArchiveManager_CloseEntry(IN handle eHandle)

설명: 엔트리 핸들을 닫는다.

(9) handle ArchiveFile_Open(IN handle eHandle)

설명: 엔트리 핸들에 해당하는 아카이브내의 파일을 연다.

반환값: 엔트리 파일 핸들

(10) int32 ArchiveFile_Read(IN handle fHandle, OUT string buffer, IN int32 len)

설명: 엔트리 파일 핸들에서 len bytes 만큼의 데이터를 읽어서 buffer에 저장한다.

반환값: 읽은 데이터 길이

(11) void ArchiveFile_Close(IN handle fHandle)

설명: 엔트리 파일을 닫는다.

(12) int64 ArchiveFile_Seek(IN handle fHandle, IN int64 offset, IN int32 option)

설명: 엔트리 파일의 포인터를 옮긴다. option 값으로는 다음 중 하나가 사용된다.

FILE_SEEK_SET	지정된 오프셋 위치로 이동
FILE_SEEK_CUR	현재의 오프셋에서 offset 값을 더함

FILE_SEEK_END	파일의 끝에서 오프셋을 계산. offset 값은 음수.

반환값: 계산되어 옮겨진 현재 오프셋

(13) int64 ArchiveFile_Tell(IN handle fHandle)

설명: 엔트리 파일 핸들의 현재 위치 포인터를 획득 한다.

반환값: 현재 위치 오프셋

(14) int64 ArchiveEntry_GetSize(IN handle eHandle)

설명: 엔트리 파일의 크기를 획득 한다. (eHandle은 엔트리 헨들임. 엔트리 파일 핸들이 아님 주의)

반환값: 엔트리 파일의 크기

(15) string ArchiveEntry_GetFilename(IN handle eHandle)

설명: 엔트리 파일의 파일명을 획득 한다. (eHandle은 엔트리 헨들임. 엔트리 파일 핸들이 아님 주의)

반환값: 저장된 파일 경로

(16) int64 ArchiveEntry_GetTimeCreate(IN handle eHandle)

설명: 엔트리 파일의 생성 시간을 획득 한다. (eHandle은 엔트리 헨들임. 엔트리 파일 핸들이 아님 주의)

반환값: CCYYMMDDhhmmss 형식의 int64형 숫자 타임스탬프

(17) int64 ArchiveEntry_GetTimeModify(IN handle eHandle)

설명: 엔트리 파일의 수정 시간을 획득 한다. (eHandle은 엔트리 헨들임. 엔트리 파일 핸들이 아님 주의)

반환값: CCYYMMDDhhmmss 형식의 int64형 숫자 타임스탬프

(18) int64 ArchiveEntry_GetTimeAccess(IN handle eHandle)

설명: 엔트리 파일의 접근 시간을 획득 한다. (eHandle은 엔트리 헨들임. 엔트리 파일 핸들이 아님 주의)

반환값: CCYYMMDDhhmmss 형식의 int64형 숫자 타임 스탬프

(19) int64 ArchiveEntry_GetOwner(IN handle eHandle)

설명: 엔트리 파일의 소유자 ID 번호를 획득 한다. (eHandle은 엔트리 헨들임. 엔트리 파일 핸들이 아님 주의)

반환값: id 값.

(20) int64 ArchiveEntry_GetGroup(IN handle eHandle)

설명: 엔트리 파일의 소유 Group ID번호를 획득 한다. (eHandle은 엔트리 헨들임. 엔트리 파일 핸들이 아님 주의)

반환값: 그룹 ID 값

(21) int64 ArchiveEntry_GetPermission(IN handle eHandle)

설명: 엔트리 파일의 퍼미션 값을 획득 한다. (eHandle은 엔트리 헨들임. 엔트리 파일 핸들이 아님 주의)

반환값: 퍼미션 값

(22) uint8 ArchiveEntry_GetType(IN handle eHandle)

설명: 엔트리 파일의 타입을 획득 한다. (eHandle은 엔트리 헨들임. 엔트리 파일 핸들이 아님 주의)

반환값: 파일 타입 값(FILE_TYPE_DIR = 디렉토리, FILE_TYPE_FILE = 일반 파일, FILE_TYPE_UNK = 기타 파일)

(23)

19. 압축 관련 함수

(1) int32 THComp_GZIP_CompressBuffer(OUT string destbuffer, IN OUT destlen, IN string srcbuffer, IN int32 srclen, IN int32 level)

설명: 입력 버퍼(srcbuffer)에서 길이(srclen) 만큼의 데이터를 압축해서 출력버퍼(destbuffer)에 저장한다. level은 압축 레벨을 지정한다.

반환값: 성공하면 0, 실패하면 0이 아닌 값을 반환 한다.

(2) int32 THComp_GZIP_UncompressBuffer(OUT string destbuffer, IN OUT destlen, IN string srcbuffer, IN int32 srclen)

설명: 입력 버퍼(srcbuffer)에서 길이(srclen) 만큼의 데이터에 대해 압축을 해제 하여 출력 버퍼(destbuffer)에 저장 한다.

반환값: 성공하면 0, 실패하면 0이 아닌 값을 반환 한다.

(3) int32 THComp_ZIP_DeflateBuffer(OUT string destbuffer, IN OUT destlen, IN string srcbuffer, IN int32 srclen, IN int32 level)

설명: GZIP 압축과 동일함. 단, ZIP 압축 알고리즘 사용.

(4) int32 THComp_ZIP_InflateBuffer(OUT string destbuffer, IN OUT destlen, IN string srcbuffer, IN int32 srclen)

설명: GZIP 압축 해제와 동일함. 단, ZIP 압축 알고리즘 사용.

(5) int32 THComp_BZ2_CompressBuffer(OUT string destbuffer, IN OUT destlen, IN string srcbuffer, IN int32 srclen, IN int32 level)

설명: GZIP 압축 해제와 동일함. 단, Bzip2 압축 알고리즘 사용.

(6) int32 THComp_BZ2_UncompressBuffer(OUT string destbuffer, IN OUT destlen, IN string srcbuffer, IN int32 srclen)

설명: GZIP 압축 해제와 동일함. 단, Bzip2 압축 알고리즘 사용.

(7)

20. 기타 함수