## Windows デバッグ講座

株式会社風太

萩原 宏紀

第1回「はじめに」

本書では、Windows のデバッグ方法について、かつてマイクロソフト株式会社でテクニ カルサポートエンジニアとして従事していた著者が解説していきます。

具体的には、Microsoft が無償提供しているデバッガを使ってプロセスや OS そのもの にアタッチしたり、クラッシュダンプを解析する事によって、OS がどの様に動いているの か、プロセスやスレッドとは何か、アプリケーションがどの様に OS と関連しているのか、 といった事が理解出来る様になると思います。

なお、本書は Windows のデバッグ方法についての解説ですので、Windows の内部アー キテクチャなどを詳細に掲載しているマニュアル本ではありません。細かい部分は割愛し、 デバッグに必要な最低限の知識を記載しております。

事前準備として、デバッガのインストールをお願いします。

昔は "Debugging Tools for Windows" という名で Microsoft からリリースされていましたが、2013 年現在は Windows SDK に同梱されています。

以下のサイトから、Windows SDK をダウンロードし、ご自身の PC にインストールして 下さい。(本書では Windows 7 x86 環境を想定しています。)

Microsoft Windows SDK for Windows 7 and .NET Framework 4 http://www.microsoft.com/en-us/download/details.aspx?id=8279

		- x
ファイル(F) 編集(E) 表示(V) お気に入り(A) ツール(T) ヘルプ(H)	uu	~~~~~
Microsoft	٩	^
Products Categories Security Support Shop		
Microsoft Windows SDK for Windows 7 and . Framework 4	NET Share 🔊 <	1
Language: English Download	Free PC updates	
The Windows SDK provides tools, compilers, headers, libraries, code samples, and a new help system that developers can use to create applications that run on Microsoft Windows.	Security patches     Software updates     Service packs     Hardware drivers     Run Microsoft Update	
Details		
System Requirements	Microsoft suggests	
	Try Visual Studio 2013 for	~

ちなみに、英語版しかありません。日本語版を探しても見つかりませんのであしからず。 [Download] ボタンをクリックすると、次の様な画面が表示されますので、[実行(R)]をク リックします。

🔿 📑 http://www.microsoft.com/en-us/download/confirmation.as; 🔎 👻 🖒 📑 Download Microsoft	Win ×		£ ☆ £
Microsoft			
Download Center			Q
Products Categories Security Support Shop			
Thank you for downloading			
Microsoft Windows SDK for Windows 7 and .NET Framework 4 If your download does not start after 30 seconds, Click here		Free PC updates	
Install Instructions		Security patches	
		Service packs     Hardware drivers	
		Bun Microsoft Update	
Popular downloads		0	
	М	crosoft suggests	
download.microsoft.com から winsdk_web.exe (497 KB) を実行または保存しますか?	R) 呆存	× fc (S) ▼ キャンセル(C)	r
	₩ <b>₩</b> ₽≥	Сару В кана + ▲ Р В л1 Ф 2	10:55

インストールが完了すると、スタートメニューから[すべてのプログラム]をクリックした時 に、「Debugging Tools for Windows (x86) 」が出現します。





ここをクリックすると、この中に "WinDbg" というアプリケーションがありますが、これ がデバッガです。

	\mu activePDF	
	🐌 Application Verifier	_ ₹:
	]] Debugging Tools for Windows (x86)	_
	😰 Debugging Help	- ケ-
	📑 Global Flags	
	Release Notes	
	🛃 Uninstall Debugging Tools for Windo	_ ⊐:
	🕎 WinDbg	
Ι	🐌 ESTsoft	,,,
	🐌 Intel 🗸 🗸	既
	▲ 前に戻る	$ $ $\sim$
	プログラムとファイルの検索 👂	21

WinDbg はユーザーモードにもカーネルモードにも使えるデバッガです。ただし、今回は この WinDbg は使いません。まずは、ユーザーモード専用デバッガを使い、ユーザーモー ドのプロセスについて勉強していきます。

その前に、ユーザーモード、カーネルモードについて少しだけ解説します。

簡単に言えば、Windows をはじめ、Linux にしても、UNIX にしても、オペレーティング システムというのは、ハードウェアを動かすためのプログラムです。そのオペレーティング システムのコア(核)の部分がカーネルモード(Linux や UNIX ではスーパーバイザーモ ードと呼んだりします。)、そして一般にサードパーティーベンダーなどが作成するアプリ ケーションプログラムはユーザーモードと呼ばれるモードで動作します。ユーザーモード のアプリケーションが、カーネルモードの関数を呼び出し、カーネルモードの関数が、ハー ドウェアにアクセスする。OS から命令を受けたハードウェアが返した値をカーネルモード が受け取り、それをユーザーモードのアプリケーションに返していく、こんな感じでコンピ ュータ上のプログラムは動いています。



ところで、デバッガはどこにインストールされているのでしょうか?

デフォルトであれば、c:¥Program Files¥Debugging Tools for Windows (x86) にインス トールされます。

では、このフォルダを開いてみて下さい。 エクスプローラーではなく、コマンドラインから開いてみて下さいね。

スタートメニューから[プログラムとファイルの検索]で、"cmd.exe" と入力して下さい。コ マンドラインが起動したら、"cd c:¥pro\*" と入力して [ENTER] キーを押して下さい。 続 いて、"cd deb\*"と入力して [ENTER]キーを押して下さい。 これでデバッガのあるフォル ダに移動します。

dir コマンドを打ってみましょう。 以下の様に出力されるはずです。

c: ¥Program Files ¥Debugging Tools for Windows (x86)>dir ドライブ C のボリューム ラベルがありません。 ボリューム シリアル番号は 2E78-B389 です c:¥Program Files¥Debugging Tools for Windows (x86) のディレクトリ 2013/11/12 23:27 <DIR> 2013/11/12 23:27 <DIR> ... 2013/11/12 22:59 <DIR> 1394 2009/08/24 14:38 71,168 adplus.doc 2010/02/01 12:27 97,040 adplus.exe 2010/02/01 12:27 29,056 adplusext.dll 2010/02/01 12:27 80,656 adplusmanager.exe 2009/08/24 14:38 2,068 adplusmanager.exe.config 2010/02/01 12:27 200,530 adplus\_old.vbs 2010/02/01 12:27 36,736 agestore.exe 2010/02/01 12:27 17,168 breakin.exe 364,816 cdb.exe 2010/02/01 12:27 2013/11/12 22:59 <DIR> clr10 2010/02/01 12:27 32,128 convertstore.exe 2010/02/01 12:27 112,512 dbengprx.exe

2010/02/01	12:27	3,557,648 dbgeng.dll
2010/02/01	12:27	1,213,200 dbghelp.dll
2010/02/01	12:27	39,184 dbgrpc.exe
2010/02/01	12:27	32,528 dbgsrv.exe
2010/02/01	12:27	151,824 dbh.exe
2010/01/08	11:07	326,336 debugger.chi
2010/01/08	11:07	5,117,792 debugger.chm
2010/02/01	12:27	419,088 decem.dll
2009/08/24	14:38	56,832 dml.doc
2010/02/01	12:27	20,864 dumpchk.exe
2010/02/01	12:27	19,840 dumpexam.exe
2010/02/01	12:27	145,168 gflags.exe
2010/02/01	12:27	362,768 i386kd.exe
2010/02/01	12:27	362,768 ia64kd.exe
2010/02/01	12:27	376,080 kd.exe
2010/02/01	12:27	34,576 kdbgctrl.exe
2010/02/01	12:27	170,256 kdsrv.exe
2009/08/24	14:38	1,196,032 kernel_debugging_tutorial.doc
2010/02/01	12:27	34,064 kill.exe
2009/09/18	11:35	10,237 license.txt
2010/02/01	12:27	80,768 list.exe
2010/02/01	12:27	28,944 logger.exe
2010/02/01	12:27	211,328 logviewer.exe
2010/02/01	12:27	365,328 ntsd.exe
2010/02/01	12:27	23,312 pdbcopy.exe
2010/02/01	12:08	2,819 redist.txt
2010/01/28	21:21	12,615 relnotes.txt
2010/02/01	12:27	69,504 remote.exe
2010/02/01	12:27	25,360 rtlist.exe
2013/11/12	22:59	<dir> sdk</dir>
2013/11/12	22:59	<dir> srcsrv</dir>
2010/02/01	12:27	92,944 srcsrv.dll
2010/02/01	12:27	30,992 symbolcheck.dll
2010/02/01	12:27	80,144 symchk.exe
2013/11/12	22:59	<dir> symproxy</dir>
2010/02/01	12:27	131,856 symsrv.dll
2009/08/24	14:38	1 symsrv.yes
2010/02/01	12:27	145,168 symstore.exe
2013/11/12	22:59	<dir> themes</dir>
2010/02/01	12:27	47,376 tlist.exe
2013/11/12	22:59	<dir> triage</dir>
2010/02/01	12:27	143,232 umdh.exe

2013/11/12 22:59 <DIR> usb 2010/02/01 12:27 139,136 usbview.exe 2010/02/01 12:27 74,512 vmdemux.exe 2013/11/12 22:59 <DIR> w2kchk 2013/11/12 22:59 <DIR> w2kfre 2010/02/01 12:27 532,752 windbg.exe 2013/11/12 22:59 <DIR> winext 2013/11/12 22:59 <DIR> winxp 51 個のファイル 16,929,054 バイト 14 個のディレクトリ 52,810,719,232 バイトの空き領域

c:¥Program Files¥Debugging Tools for Windows (x86)>

この中に cdb.exe というモジュールがあります。これが、コマンドライン形式のユーザー モード専用デバッガです。一方、kd.exe というモジュールがカーネルモード専用デバッガ です。そして、よく使うツールとして tlist.exe が同梱されています。

## tlist を実行してみて下さい。

c:¥Program Files¥Debugging Tools for Windows (x86)>tlist

- 0 System Process
- 4 System
- 264 smss.exe
- 340 csrss.exe
- 388 wininit.exe
- 396 csrss.exe
- 444 winlogon.exe
- 476 services.exe
- 492 lsass.exe
- 500 lsm.exe
- 616 svchost.exe
- 704 svchost.exe
- 768 MsMpEng.exe
- 860 svchost.exe
- 892 svchost.exe
- 932 svchost.exe
- 988 svchost.exe
- 1104 svchost.exe
- 1284 svchost.exe
- 1400 wlanext.exe

. . .

tlist.exe は、現在実行中のユーザーモードプロセスを表示します。左の数字はプロセスに 紐づけられるプロセス ID です。右は実行モジュール名です。

それでは、メモ帳を起動してみて下さい。

スタートメニューから[プログラムとファイルの検索]で、"notepad" と入力して下さい。メ モ帳が起動したら、コマンドラインに戻り、tlist をもう一度実行してみて下さい。今度は 以下の様に出力されるでしょう。

3684 notepad.exe 無題 - メモ帳 3944 tlist.exe c:¥Program Files¥Debugging Tools for Windows (x86)>

tlist の結果に、今起動したメモ帳が表示されたかと思います。 ちなみに、プロセス ID は固定ではありません。起動する度に OS が適当に振るので、こ うやって都度 tlist でプロセス ID を調べる必要があります。(注:一度起動されたプロセ スのプロセス ID は、そのプロセスが終了するまでは同じ ID です。)

さて、ようやくデバッガの起動です。先程起動したメモ帳をデバッグしてみましょう。 cdb.exe の引数に -p をつけ、notepad.exe のプロセス ID を入力します。 例えば、上のプロセス ID の例でいえば、

"cdb -p 3684"

となります。これを入力してみると、

. . . .

(e64.16a0): Break instruction exception - code 80000003 (first chance)
eax=7ffdb000 ebx=00000000 ecx=00000000 edx=773ef1d3 esi=00000000 edi=00000000
eip=77384108 esp=02aefce4 ebp=02aefd10 iopl=0 nv up ei pl zr na pe nc
cs=001b ss=0023 ds=0023 es=0023 fs=003b gs=0000 efl=00000246
ntdll!DbgBreakPoint:
77384108 cc int 3
0:004>

となって止まるはずです。この時、メモ帳はどうなっていますか?おそらく、クリックして も、ウンともスンとも言わないはずです。この状態をブレイクと言います。つまり、cdb を 使って、notepad.exe をデバッグできる状態に停止した事になります。これを、デバッガ でアタッチ (attach) したと言います。この状態で "qd" と入力すると、デバッガが notepad.exe を解放し、メモ帳は操作可能な状態に戻ります。これをデタッチ (detach) と言います。



## では、もう一度アタッチしてみて下さい。

そして、"~0kb" と入力してみて下さい。

0:004> ~0kb

ChildEBP RetAddr Args to Child WARNING: Stack unwind information not available. Following frames may be wrong. 0027f964 0020148a 0027f988 0000000 00000000 ntdll!KiFastSystemCallRet 0027f9a4 002016ec 00200000 0000000 000b23da notepad+0x148a 0027fa34 764bed5c 7ffdf000 0027fa80 773b37eb notepad+0x16ec 0027fa40 773b37eb 7ffdf000 77d0196d 00000000 kernel32!BaseThreadInitThunk+0x12 0027fa80 773b37be 00203689 7ffdf000 00000000 ntdll!RtlInitializeExceptionChain+0xef 0027fa98 0000000 00203689 7ffdf000 00000000 ntdll!RtlInitializeExceptionChain+0xc2

"~0" とは、0 番のスレッドを指定、"kb" とはスタックバックを表示するコマンドです。 つまり、notepad.exe というプロセスで動いている複数のスレッドのうち、0 番のスレッ ドのスタックバックを表示しなさい、という命令が "~0kb" となります。

スタックバックですが、あるモジュールの関数が次のモジュールの関数を呼ぶ度に、スタッ クというメモリ領域にその履歴を記録していきます。スタックの名の通り、これは下から順 に積まれていきますので、見る時も下から見ていきます。上の例でいうと、ntdll というモ ジュールの RtlInitializeExceptionChain という関数から ntdll というモジュールの RtlInitializeExceptionChain を呼び、更に、 kernel32 というモジュールの BaseThreadInitThunk を呼び、そこから notepad.exe を呼ぶ…という感じです。 ただ、notepad+0x16ec とか書かれていても、何の事かわかりませんよね?

では、

".sympath srv\*c:¥localsymbols\*http://msdl.microsoft.com/download/symbols" と入力してみて下さい。

0:004> .sympath srv\*c:¥localsymbols\*http://msdl.microsoft.com/download/symbols Symbol search path is: srv\*c:¥localsymbols\*http://msdl.microsoft.com/download/symbols Expanded Symbol search path is: srv\*c:¥localsymbols\*http://msdl.microsoft.com/download/symbols

その後、

".reload"

と入力します。

0:004> .reload Reloading current modules

そして、もう一回、"~0kb" と入力してみて下さい。

どうなりましたか?

今度は、以下の様に notepad の横に関数名が出てきたかと思います。

0:004> ~0kb

ChildEBP RetAddr Args to Child

0027f944 7707cde0 7707ce13 0027f988 0000000 ntdll!KiFastSystemCallRet 0027f948 7707ce13 0027f988 0000000 0000000 USER32!NtUserGetMessage+0xc 0027f964 0020148a 0027f988 0000000 0000000 USER32!GetMessageW+0x33 0027f9a4 002016ec 00200000 0000000 000b23da notepad!WinMain+0xe6 0027fa34 764bed5c 7ffdf000 0027fa80 773b37eb notepad!\_initterm\_e+0x1a1 0027fa40 773b37eb 7ffdf000 77d0196d 00000000 kernel32!BaseThreadInitThunk+0xe 0027fa80 773b37be 00203689 7ffdf000 0000000 ntdll!\_\_RtlUserThreadStart+0x70 0027fa98 0000000 00203689 7ffdf000 0000000 ntdll!\_RtlUserThreadStart+0x1b これは、何をしたのでしょうか?

実は、Microsoft の提供するシンボルサーバーという所にアクセスして、シンボルファイル というファイルをダウンロードしたのです。

エクスプローラーで c ドライブを見ると、

c:¥localsymbols

というフォルダが出来ていると思います。この中に、必要なシンボルファイルがダウンロー ドされており、そのおかげで関数名が出てくるようになったのです。

ここまでで、何となく推測できると思いますが、このシンボルファイルはモジュールごとに 作られ、このモジュールのこの関数は、どこのアドレスから始まっていますよ、という情報 が格納されているのです。

ただ、いちいちデバッガ上で、シンボルの場所を入力するのは面倒ですよね?

そんな人のために、デバッガには環境変数を設定する事が出来ます。

スタートメニューからコントロールパネルを開き、[システムとセキュリティ] - [システム]-[設定の変更]をクリックし、システムのプロパティを開きます。

(もしくはコマンドラインから、"control sysdm.cpl" でも結構です。)

システムのプロパティ	×
コンピューター名 ハードウェア 詳細設定 システムの保護 リモート	_
次の情報は、このコンピューターをネットワーク上で識別するために使われます。	
コンピューターの説明( <u>D</u> ):   (例本) 「 (の本) 「 (の本) 「 (の本) 「 (の本) 」 (の本) 」 (の本) 」 (の本) 「 (の本) 」 (の本) (の本) (の本) 」 (の本) (の本) (の本) ( (の本) (の本) (の本) ( (の本) (の本)	
7ルコンピューター名: CF-Y8	
ワークヴループ: WORKGROUP	
ドメインまたはワークグループに参加するためのウィザードを使用 するには 「ネットワーク ID」をクリックしてください。	
コンピューター名を変更したりドメインに参加したりするには (変) 更]をクリックしてください。	
OK キャンセル 適用(A)	

システムのプロパティが開いたら、[詳細設定]タブを開き、下の[環境変数(N)...]ボタンを クリックします。

システムのプロパティ	×
コンピューター名 ハードウェア 詳細設定 ミステムの保護 リモート	_
Administrator としてログオンしない場合は、これらのほとんどの変更はできません。	
パフォーマンス 視覚効果、プロセッサのスケジュール、メモリ使用、および仮想メモリ	
設定(S)	
ユーザー プロファイル ログオンに関連したデスクトップ設定	
設定( <u>E</u> )	
一起動と回復 システム記動、システム障害、およびデバッグ情報	
設定(D	
環境変数( <u>N</u> )	

変数	值
TEMP	%USERPROFILE%¥AppData¥Local¥Temp
TMP	%USERPROFILE%¥AppData¥Local¥Temp
	新規( <u>N)</u> 編集( <u>E</u> ) 削除( <u>D</u> )
ステム環境変数( <u>S</u> ) 変数	
ステム環境変数( <u>S</u> ) 変数 asling	新規(N)… 「編集(E)… 」 削除(D) 値 Destination=file
ステム環境変数( <u>S</u> ) 変数 asl.log CLASSPATH	新規(N) 編集(E) 削除(D) 值 Destination=file .C¥Program Files¥QuickTime¥QTSystem¥QT
ステム環境変数( <u>S</u> ) 変数 asl.log CLASSPATH ComSpec	新規(N) 編集(E) 削除(D) 値 Destination=file ;C¥Program Files¥QuickTime¥QTSystem¥QT C:¥Windows¥system32¥cmd.exe
ステム環境変数( <u>S</u> ) 変数 asl.log CLASSPATH ComSpec FP NO HOST CHF	新規(N) 編集(E) 削除(D) 値 Destination=file ;C:¥Program Files¥QuickTime¥QTSystem¥QT C:¥Windows¥system32¥cmd.exe NO

下の[システム環境変数(S)]の[新規(W)...]ボタンをクリックし、

[変数名(N):]に

"\_NT\_SYMBOL\_PATH"、

[変数値(V):]に

"SRV\*c:¥localsymbols\*http://msdl.microsoft.com/download/symbols" を入力し、[OK] をクリックします。

システム変数の編集	
変数名( <u>N</u> ):	_NT_SYMBOL_PATH
変数値(⊻):	SRİV*c:¥localsymbols*http://msdl.microsoft.com.
	OK キャンセル

すると、[システム環境変数(S)]の所に、\_NT\_SYMBOL\_PATH が設定されると思います。 それが確認できたら、[OK]をクリックしてシステムのプロパティを閉じます。

36.6X	値			
TEMP	%USERPROFILE%¥AppData¥Local¥Temp			
ТМР	%USERPROFILE%¥AppData¥Local¥Temp			
	新規( <u>N)</u> 編集(E) 削除(D)			
ステム環境変数( <u>S</u>	)			
ステム環境変数( <u>S</u> 変数	) 値			
ステム環境変数(S 変数 _NT_SYMBOL_PA	) 値 TH SRV*c¥localsymbols*http://msdl.microsoft.co			
ステム環境変数(S) 変数 _NT_SYMBOL_PA asl.log	) 値 TH SRV*c¥localsymbols*http://msdl.microsoft.co Destination=file			
ステム環境変数( <u>S</u> 変数 _NT_SYMBOL_PA asi.log CLASSPATH	) 値 TH SRV*c¥localsymbols*http://msdl.microsoft.co Destination=file ;C:¥Program Files¥QuickTime¥QTSystem¥QT			
ステム環境変数( <u>S</u> 変数 <u>NT_SYMBOL_PA</u> asl.log CLASSPATH ComSpec	) (値 TH SRV*c¥localsymbols*http://msdl.microsoft.co Destination=file ;C¥Program Files¥QuickTime¥QTSystem¥QT C¥Windows¥system82¥cmd.exe			

以降、PC を再起動した後にはこの環境変数が自動で設定されますので、デバッガを動かしたとき、自動でシンボルサーバーから必要な OS のシンボルをダウンロードしてくれます。

OS のシンボルファイルはこれで良いのですが、その他のアプリケーション用のシンボルフ ァイルは別途、アプリケーション提供元から入手し、.sympath で指定して読み込む必要が ありますので、注意して下さい。 あと、デバッガ用のコマンドラインのショートカットをデスクトップに作成しておくと便利です。コマンドラインのモジュールは

C:¥Windows¥System32¥cmd.exe

ですので、エクスプローラで c:¥Windows¥System32 に移動し、cmd.exe 上で右クリッ クしてショートカットを作成します。

, 퉬 « ローカル ディスク (	(C:) > Windows > Syste	em32	• •	System32	の検索	٩
■ 開く 書き込む	新しいフォルダー				III 🔹 🗖	0
□□ 開く 書き込む に入り ウンロード スクトップ 近表示した場所 :yDrive ブラリ キュメント クチャ デオ ュージック ムグループ	新しいフォルター 名前 ③ clb.dll ④ clbcatq.dll ④ clbcatq.dll ④ clcanmgr.exe ④ clfs.sys ④ clfsw32.dll ④ cliconfg.dll ④ cliconfg.rll ■ clip.exe ④ clusapi.dll ④ cmcfg32.dll ■ cmd.exe ⑧ cmdial32.dll ■ cmdkey.exe		調       調       2	更新日時 2009/07/14 10:15 2009/07/14 10:15 2009/07/14 10:14 2009/07/14 10:26 (A) (A) (Y) (Q) cy Essentials でスキ	■== ▼ ■ 種類 アプリケーショ… アプリケーショ… アプリケーション システム ファイル ・ ・ ・	
ビューター ーカル ディスク (C:)	ा cmdxey.exe ड्री cmdl32.exe छि cmicryptinstall.dll छि cmifw.dll		タスク ハーに表示 スタート メニュー 以前のバージョン 送る(N)	<する(K) -に表示する(U) の復元(V)	•	
キュリティで保護された rokiのiPhone	icom in the important of the importa		ンり(T) 切り取り(T) コピー(C)			2 +
Und.exe 更新日日 アプリケーション サイス	ನ: 2010/11/20 4:17 ನ: 295 KB		ショートカットの	作成(S)		

これをデスクトップに移動し、右クリックしてプロパティを開きます。プロパティを開いた ら、[ショートカット] タブの [作業フォルダー(s)]を"C:¥Program Files¥Debugging Tools for Windows (x86)"とします。こうする事で、このショートカットを起動した直後 からカレントディレクトリは、"C:¥Program Files¥Debugging Tools for Windows (x86)" となります。

m Debugのプロパティ
画面の色         互換性         セキュリティ         詳細         以前のバージョン           全般         ショートカット         オブション         フォント         レイアウト
Debug
種類: アプリケーション
場所: System32
リンク先( <u>」</u> ): C.¥Windows¥System32¥cmd.exe
作業フォルダー(S): [¥Program Files¥Debugging Tools for Windows (x86)"] ショートカット なし
キャット 実行時の 大きざ( <u>B</u> ):
コメント(O): ファイルの場所を開く(E) アイコンの変更(C) 詳細設定(D)
OK キャンセル 適用(A)

次に、[オプション]タブの [編集オプション] で [簡易編集モード(Q)] にチェックを入れ ます。この [簡易編集モード(Q)] にチェックを入れる事で、コマンドライン上でカット& ペーストなどが使用出来る様になります。

쮥 Debugのプロパティ		<b>×</b>
画面の色         互換性           全般         ショートカ	セキュリティ ット オプション	詳細 以前のバージョン フォント レイアウト
- カーソルのサイズ ● <u>「「(S)</u> ● 中(M) ● 大(L) - コマンドの履歴 バッファー サイズ( <u>B</u> ): バッファーサイズ( <u>B</u> ): バッファー数( <u>N</u> ): ■ 重複する古い履歴 現在のコードページ(( <u>932</u> (ANSL/OEL	500 4 全破桒(D) 2) 4 - 日本語 Shift-JIS)	<ul> <li>表示オブション</li> <li>● ウィンドウ表示(Ψ)</li> <li>● 全画面表示(Ψ)</li> <li>編集オブション</li> <li>✓ 簡易編集モード(Q)</li> <li>✓ 挿入モード(I)</li> </ul>
	OK	<b>キャンセル</b> 適用( <u>A</u> )

あと、[フォント]タブ、[レイアウト] タブは好みで変更します。(ちなみに、私は [フォント] タブでは [6×13]、[レイアウト] タブでは、[画面バッファーのサイズ] で [幅(W):] 120 [高さ(H):] 9999 とし、[ウィンドウのサイズ] で [幅(I):] 120 [高さ(E):] 40 とし ます。)

ז Debugのプ	ロパティ			×			
画面の色	互換性	セキュリティ	洋糸田	以前のバージョン			
全般	ショートカット	オプション	フォント	レイアウト			
ウィンドウのブ	サイズ( <u>S</u> )						
3×5							
$6 \times 13$							
10 x 22							
12 × 27							
			-				
			· ·				
フォント( <u>F</u> )	□太字フォン	ノト( <u>B</u> )					
<b>査</b> MS ゴシ	ック						
ラスター フォン	15						
一選択したフォ	ント:Terminal —	zn	ギれの文字()	t.			
C:¥WINDOWS> dir 6ピクセルの幅							
SYSTEM <dir> 10-01-99 13 ピクセルの高さ</dir>							
OK 「キャンセル」 適用( <u>A</u> )							

<ul> <li>ア Debugのプロパティ</li> <li>         画面の色 互換性 セキュリラ         全般 ショートカット オブ     </li> </ul>	Fィ 詳細 レ ション フォント	・ 、 前のバージョン レイアウト
ウィンドウのプレビュー	画面バッファーのサイズ 幅(W): 高さ(日): ウィンドウのサイズ ─ 幅(D): 高さ(E): ワインドウのロロ画 左から(L): 上から(T): ▼ システム設定を使う	120 9999 120 40 75 75 75 (P)
ОК	<b>*</b> #>\triangle #	適用( <u>A</u> )

これで、準備は完了です。[適用(A)]をクリックして、プロパティ画面を閉じます。

では、今回のまとめです。



今回は、盛りだくさんの内容でしたが、いかがでしたでしょうか?

おそらく、ここまででユーザーモードのプロセスについてアタッチ、デタッチ、スタックバックの確認は出来る様になったと思いますので、次回はもう少し深く見ていきます。