コンピュータメッセージ考

～多言語対応と簡単なコンピュータ英語入門～

Ver. 1.1

リナックスアカデミー矢越昭仁

2012/11/10

コンピュータが発するメッセージを題材にして、そこで用いられるコンピュータ用語（英語）や、そもそも言葉をどうやって扱うのかを解説します。ますますグローバル化が叫ばれる世の中で、世界で通用するエンジニアになるためのきっかけになれば幸いです。

目次

はじめに

IT 業界において技術的な知識とは別に最も重要なのは「飽くなき探究心＝興味」だといわれています。この IT 特別講座では皆さんの知識・技術の向上を促し、「飽くなき探究心＝興味」を満足させる講座を提供することを目的としています。講座中は遠慮なく質問し、より理解を深めるとともに、新たな疑問は次の講座の開催要望として意見をください。

表記について

この資料では以下の表記としています。

・フォント

コンピュータの操作および設定ファイルはクーリエフォント(タイプライター風)を用います。

```
search t123006.la.net
nameserver 10.20.123.6
```

・プロンプト

コマンド入力例がある場合は、先頭はプロンプト($または#)で始めます。

$ は一般ユーザでの操作、#はルートユーザでの操作を表します。なおユーザ切り替え(su)の表記は省略しています。

・強調（ボールド）

コマンド入力では、キーボードから入力する場合を、設定ファイルの場合は修正箇所など特に強調したい場合に**ボールド**を使います。

```
$ date
Mon Mar 5 12:32:41 JST 2012
```

```
DEVICE=eth0
NM_CONTROLLED=yes
ONBOOT=yes
```

・凡その作業時間

凡その作業時間とは、過去に同様の作業を経験した人が再度実行した場合にかかる時間を想定しています。つまり事前調査や試行錯誤の時間を含まない作業時間を指します。

オンラインバックアップ

矢越が実施した IT 特別講座の資料（補足資料、例題等含む）は、以下の URL にて掲載しています。この URL はリナックスアカデミー会員限定となっていますので、それ以外への再配布・再掲載は遠慮ください。

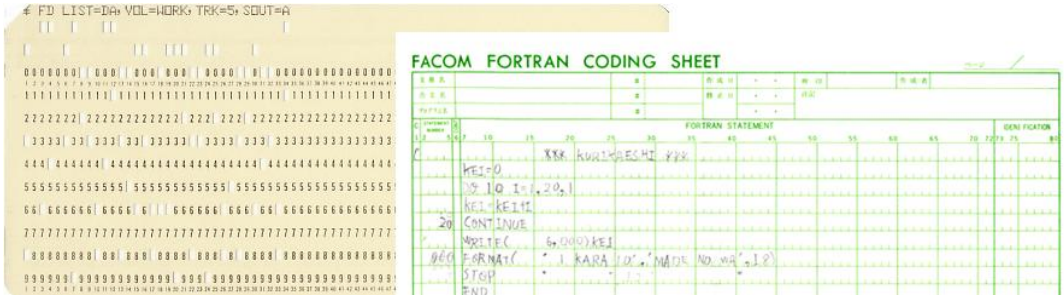
http://ycos.sakura.ne.jp/LA

また講座・資料への質問、要望は下記までメールをお願いします。

ycos001@yahoo.co.jp

コンピュータとの対話

その昔、コンピュータと直接やり取りする事を「対話:Conversation」と呼んでいました。当時は必要なデータやプログラムを紙カードやフロッピーに用意。オペレータに手渡すと、彼らがコンピュータ室でシステムを操作し処理してくれます。結果は印刷された帳票で、利用者が直接コンピュータを触る事はありませんでした。



その後、TSS システム(Linux の原型)が登場し、直接コンピュータを操作するようになります。キーボードからコマンドを入力すると、ディスプレイにその結果が表示されます。利用者はそのメッセージに従い間違いを修正したり、データを入力したりしていました。このやり取りが対話です。

現在、「コンピュータとの対話」は、日常茶飯事になっています。PC を操作していると、いろんなメッセージが表示されます。時には CD をセットしろ、再起動するからファイルを保存せよなど、人の方がこき使われる場合さえあります。
そうした対話も意味は通ってこそ成り立ちますが、しばしば難解なメッセージに遭遇することもあります。「#DIV/0!」といった記号や、英語で「No such file or directory」と表示されると少し躊躇する人もいるでしょう。また場合によっては、文字化けを起こし「繝峨Γ繧、繝ウ蜷搾シ」と全く意味不明な場合もあります。

この講座では、コンピュータが発するメッセージを題材にして、そこで用いられるコンピュータ用語(英語)や、どのように人間の言葉を扱うのかを解説します。ますますグローバル化が叫ばれる世の中で、世界で通用するエンジニアになるために必要な「教養」としての知識を紹介します。

なぜ英語なのか

もともとコンピュータは米国製で他の言語を扱うように設計されていませんでしたし、基本的に計算が目的でした。世界初のコンピュータ言語は FORTRAN で 1950 年代に登場しました。FORTRAN の名前の由来は Formula Translation で、「数式をコンピュータ言語に変換する」という意味です。
数式だけであれば、特に母国語を気にする必要もありませんでした。(英語風な記号の羅列)

例) FORTRAN プログラム例(一部)

```
C  SAMPLE PROGRAM FOR FORTRAN III
  101    FORMAT (3I5)
         READ INPUT TAPE 5, 101, IA, IB, IC
         IX = IA + IB + IC
         IF (IX) 210, 220, 230
         STOP 9
  210    WRITE OUTPUT 6, 211, IX
  211    FORMAT (7HPLUS : ,I5)
```

その後 1960 年に登場し、今でも日本の特に金融業界で使われている COBOL(COmmon Business Oriented Language、商業向け共通言語)は、より英語に近い表現になっています。そもそも FORTRAN では人が理解しづらいという事で、英文法に近い表記を目指したのでした。
現在はあらゆるところでコンピュータが利用され、一般の利用者向けには日本語化が進んでいます。

技術的な制約

1980 年代、IBM PC がブレークし、それまで個人の趣味用であった PC がビジネスで使える事が立証されました。IBM PC はその技術を公開(Open Architecture)し、市場が活性化し短期間に世界標準機となりました。
しかし、日本だけは例外でした。国内 PC 市場は完全なガラパゴス化に陥っており、しかも NEC の PC-9800 シリーズが国内市場の 90%を抑えるという極端な寡占状態でした。これは日本語、とくに漢字を扱う機能が当時の IBM PC になかったためです。
当時は英語以外の言語を扱えるコンピュータは、日本ぐらいしかなく多くの国では英語のまま利用していました。この事もコンピューター＝英語に拍車をかけるものでした。

実際、母国語が使えなくても問題ない場合があります。たとえば電卓などは、四則演算など必要最小限の機能は説明書を見なくても使えます。また表示されるのは数字だけですので、英語や日本語の差異はありません。

英語が世界標準言語となっている事実

英語が国際共通語として用いられているのは周知の事実だと思います。古くは大英帝国に端を発し、金融を中心とした国際市場における米英の影響力は絶大です。
さらに IT の研究開発は米国中心に進み、多くの技術は米国から発信されています。また発信される量も膨大で、最近では母国語に翻訳する事が少なくなりました[1]。

例えば、Program（code）は算符、Data は算量、Algorithm は算法といった訳がありましたが、最近ではデプロイ(Deploy, サーバへのプログラムやライブラリの配備)、プロビジョニング(Provisioning, コンピュータ資源の割当予約と引当)、ビッグデータ(Big data,膨大なデータ処理)などカタカナ表記がほとんどです。

母国語でない人たちを含めると英語を使う人口は 14 億人以上と多く、IT 業界、とくにインターネット上では標準語となっています。

言語人口（母国語、公用語） source: 2009 Ethnologue 16th

順位	言語	人口(M)	順位	言語	人口(M)
1	中国語	1,213	6	ベンガル語	181
2	スペイン語	329	7	ポルトガル語	178
3	英語	328	8	ロシア語	144
4	アラビア語	221	9	日本語	122
5	ヒンディ語	182	10	ドイツ語	90

以上の事から、このコースでは英語で表示されるコンピュータメッセージのうち、特徴的なものを解説し理解の手助けを行います。

[1] 最初で最後のコンピュータ辞典 岩波「情報科学事典」1990 年刊

コンピュータでの多言語対応

すでに解説したように、コンピュータの世界では英語を用いるのが一般的です。しかしそれは、プログラムの作成者や、システムの設計者（さらに言えば、国内では先端を走る人々）の話です。
利用する人たちから見れば、普段使っている言語が扱える事が必須です。
ここではコンピュータの黎明期からどうやって日本語が扱えるようになったのか、また日本語以外はどうやって扱われるのかを解説します。

アルファベットの世界

初めてコンピュータに文字を持ち込んだ際、文字に連番を付けて管理をしました。1963 年米国で規格化された ASCII (American Standard Code Information Interchange)がそれです。1 文字を 1 オクテットで表し、英数字、特殊文字および制御コードが表現できます。当時は通信の品質が低く、1 文字ごとにエラー検出のためのパリティビットが用意されていたため、実際に割当が可能だったのは 0～127 の 128 種類（7bit）でした。同じころ欧州では ECMA-1(6,7bit)が制定されました。

これらをうけ、ISO が 1967 年に ISO R 646 として欧米のラテン文字の標準を制定しました。この規格では欧州や日本などの国ごとに自由に設定できる枠を設けていたため、同じ番号でも異なる文字を表すという問題がありました。

コード (16 進)	ASCII (US)	DE	DK NO	GB	HU	JP	MT	SE	YU
23	#	#	#	£	#	#	#	#	#
24	$	$	$	$	¤	$	$	¤	$
40	@	§	@	@	Á	@	@	@	Ž
5B	[	Ä	Æ	[	É	[	ġ	Ä	Š
5C	\	Ö	Ø	\	Ö	¥	ż	Ö	Đ
5D	]	Ü	Å	]	Ü	]	ħ	Å	Ć
5E	^	^	^	^	^	^	^	^	Č
60	`	`	`	`	á	`	ċ	`	Ž
7B	{	ä	æ	{	é	{	Ġ	ä	Š
7C	|	ö	ø	|	ö	|	Ż	ö	Đ
7D	}	ü	å	}	ü	}	Ħ	å	Ć
7E	~	ß	~	~	˝	¯	Ċ	~	Č

（ISO 国コード） US:米国 DE:ドイツ、DK:デンマーク、NO:ノルウェー、GB:英国、HU:ハンガリー、JP:日本、MT:マルタ、SE:スェーデン、YU:ユーゴスラビア(マケドニア)

また、当時の出力装置はタイプライターが全盛でした。フランス語などは 2 重に印字して合成可能だったため、特別なコード表を用意する事はありませんでした。（1991 年に改訂され ISO646 となり、ASCII と完全互換になりました。）
日本のキーボードは ISO 規格に準じているため、バックスラッシュではなく円記号（¥）が採用されています。

半角カナ

ISO 646 を用いても、日本語を扱うには文字数が全く不足しています。仮名だけでも 50 文字、常用漢字は 2,136 文字もあります。そこで、カタカナの集合を規定した JIS C6220 を 1969 年に公開します。カタカナ（一般、拗音・促音で使う小文字、句読点、カッコを含む）62 文字を定めています。
7bit で運用する場合は、英数字の部分と、カナ部分で表を切り替えるための SI(Shift-In、15(0F))、SO(Shift-Out、14(0E))が必要となります。8bit で運用する場合は、8bit 目が 1 で始まる領域に割り振られています。
最近は減りましたが、厳密な規定では電子メール(SMTP)では 7bit で運用されるため、インターネットを経由したメールで日本語が化ける場合があります。
例)「ABC アイウ」を 7bit, 8bit で表現した場合のデータ並び。

8bit	41	42	43		B1	B2	B3	
7bit	41	42	43	0F	B1	B2	B3	0E

なお半角という言葉は本来印刷用語でしたが、ワープロ専用機が全盛のころに普及しました。原稿用紙ように 1 文字を 1 つの枠に書く場合を「全角」としました。アルファベットを同じように表すと間延びして見えるため、1 つの枠に 2 文字を表しました。ちょうど全角の半分の幅であることから「半角」と呼ばれるようになりました。

漢字の扱い

1978 年代の終盤になると、コンピュータでも漢字を扱う必要が高まり、JIS が漢字コードを取りまとめる事になります。当時はまだ漢字を扱うプリンターも少なく、活字印刷以外では和文タイプライターが主流でした。(日本初のワードプロセッサは 1978 年に東芝が発売しています)。
1978 年 JIS C6626 として第 1 水準(2,965 字)、第 2 水準(3,384 字)、非漢字(453 字)が公開されます。実際には JIS だけで検討したため、後に国文学者から誤字であると指摘をうけたり、選定基準について疑問視されるなど、幾度か改訂が行われています。特に 1997 年に行われた改訂では、補足として選定方法も言及するなど、選定プロセス自体も重要となっているようです。
2006 年の改訂では、第 1 水準(2,965 字)、第 2 水準(3,390 字)、第 3 水準(1,259 字)、第 4 水準(1,183 字)の合計 11,233 字が選定されています。

JIS 漢字コードでは、半角 2 文字分の 2 バイトを使って表しています。実際には 7bit 半角と互換を持たせるため、7bit × 7bit の範囲に収められています。つまり最大でも 128×128＝16,384 文字の領域しか使いません(下図、グレーの部分)。さらに制御コード用に予約された領域があるため、実際の文字に割り当てているのは、126×126 の領域となっています(下図「JIS 漢字で有効な領域」)。

<table>
<tr><th>下位
上位</th><th>00,01.</th><th>02,03,…7E,7F</th><th>80,81</th><th>82,83…FE,FF</th></tr>
<tr><td>00
01</td><td></td><td></td><td></td><td></td></tr>
<tr><td>02
03
:
7E
7F</td><td></td><td>JIS 漢字で
有効な領域</td><td></td><td></td></tr>
<tr><td>80</td><td></td><td></td><td></td><td></td></tr>
<tr><td>81
:
9F</td><td colspan="2">Shift JIS の領域</td><td colspan="2"></td></tr>
<tr><td>A0
:
DF</td><td></td><td></td><td colspan="2">EUC の領域</td></tr>
<tr><td>E0
:
FF</td><td colspan="2">Shift JIS の領域</td><td colspan="2"></td></tr>
</table>

JIS 漢字では、半角文字(ASCII)と全角文字の切り替えにも、制御文字を用いています。たとえば「漢字 AB」であれば、データは以下の並びになります。

文字	Kin(漢字開始)			漢		字		Kout(半角開始)			A	B
コード(16 進)	1B	24	42	34	41	3B	7A	1B	28	42	41	42
7bit 文字	ESC	$	B	4	A	;	z	ESC	(	B	A	B

半角と全角が混在すると頻繁に Kin/Kout が挿入され、データ量が大きくなる事が分かると思います。

処理能力が低かった初期のパソコンでは、半角文字で使わない領域を1バイト目に割当て、2 バイト目をすべて使う変則的な移動を行う事で、この問題を解決しました。これがシフト JIS と呼ばれる漢字コード体系です。

また UNIX の世界では、日本を中心に独自の拡張を行い漢字が使えるようにしました。これを EUC(Extended UNIX Code)と呼びます。

文字	漢		字		A	B
Shift JIS	8A	BF	8E	9A	41	42
EUC-JP	B4	C1	BB	FA	41	42

この様に日本語を表現するためのコード体系はとても複雑で、よく文字化けを起こしていました。プログラムが与えられたデータを、どのコードで表示するかを明確にする方法が確立されておらず、混乱したためです。
またプログラム言語では、7bit 文字にしか対応しておらず、2 バイト以上の文字を上手く扱えないものも多数存在します。C や Perl なども問題を抱えています。

日本語に限らず、世界の言葉を共通のコードで扱おうという流れが 1990 年頃から起こり、現在では Unicode と呼ばれる規格になりました。Windows Vista, Linux 2.6 といった OS や、データベース、Java や PHP といったプログラミング言語にも採用され、多言語に対応しています。
ちなみに Unicode では CJK 漢字と呼ばれ、中国語・日本語・韓国語で統一した漢字コードを用いています。

文字	漢			字			A	B
Unicode	E6	BC	A2	E5	AD	97	41	42

システムの国際化

さて、他バイト文字を扱うだけでは各国の利用者が使える状態にはなりません。表記の順序や、単位などを含め、いろいろな標準に則りユーザーインターフェースを提供する必要があります。OS やプログラミング言語などを、世界中で利用できるようにすることを i18n(internationalization)と呼びます。
各国の個別対応は、L10N(localization)と呼び、i18n の上に複数の L10N を載せることも可能です。
Linux や iPhone などでは、簡単に言語を変更・追加する事ができます。

L10N	日本	米国	中国	韓国	… 国別の機能
i18n	共通機能として提供				

Linux や Internet の世界では、この L10N を locale(ロケール)と呼んでいます。Linux の locale のマニュアルを参考に、機能を一覧化すると以下のようになります。

環境変数(機能名)	動作
LC_COLLATE	文字をソートする場合の順序の規定。例えばドイツ語の「ß」は「ss」と見なす。
LC_CTYPE	文字操作の規定。文字数を数える、大文字・小文字変換など。
LC_MONETARY	数字(特に金額)の桁取りと、小数点。例えばフランスでは桁取りがピリオド(.)で、小数点がカンマ(,)
LC_MESSAGES	メッセージ表示の言語、および肯定・否定の表現の規定。
LC_NUMERIC	数値の扱い。特にキーボードから入力された文字列の判定規則。
LC_TIME	時間の表現。欧州や日本では 24 時間、米国では 12 時間表示など。
TZ	タイムゾーン。国際標準時(UTC)からの時差と、サマータイムの有無やその期間。(ロケールコードではなく、Time Zone を指定)

環境変数に、ロケールコードを指定し L10N 機能を有効化します。ロケールコードは
言語コード __ 国コード . 文字コード
となります。言語コード、国コードとも ISO で規定されています。
例)

ja_JP.UTF-8	日本の日本語で、Unicode(UTF-8)
ja_JP.shift_jis	日本の日本語で、Shift JIS
en_US	米国の英語
en_GB	英国の英語

Linux でディスクの空き容量をチェックした際の、言語ごとの比較
（$ df –h の実行結果）

ja JP.UTF-8	fr FR
Filesystem サイズ 使用 残り 使用% マウント位置 /dev/mapper/VolGroup00-LogVol00 7.2G 5.5G 1.3G 82% / /dev/sda1 99M 38M 57M 40% /boot tmpfs 125M 0 125M 0% /dev/shm	Sys. de fich. Tail. Occ. Disp. %Occ. Monté sur /dev/mapper/VolGroup00-LogVol00 7,2G 5,5G 1,3G 82% / /dev/sda1 99M 38M 57M 40% /boot tmpfs 125M 0 125M 0% /dev/shm

フランスのロケールでは、ヘッダがフランス語、小数点はカンマ(,)になっている。

メッセージの部分を別ファイルとして用意し、ロケールでそれらを指定するという方法を使っています。有名なツールに gettext があり、多様なプログラミング言語のソースコードを修正しメッセージ部分を取り出します。翻訳担当が最初に作成されたメッセージファイルを元に翻訳し、必要な言語のメッセージファイルを用意します。

CentOS では、/usr/share/locale/言語/LC_MESSAGES/ にメッセージファイルが格納されます。
オンラインマニュアルは /usr/share/man 下に英語版と言語別ファイルが格納されています。

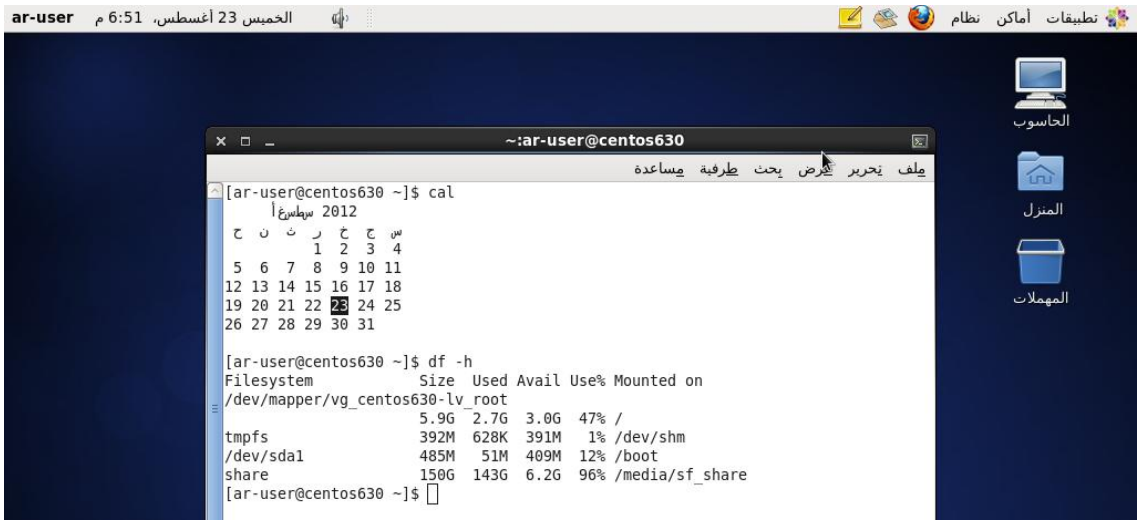
```
/usr/share/man/
  |-- man1
        :
  |-- man8
  |-- ja
  |   |-- man1
        :
  |   `-- man9
  |-- ko
  |   |-- man1
        :
      `-- man9
```

GUI の i18n

GUI での i18n は更に面白いものとなります。以下に変わった言語のサンプル(CentOS 版)を掲載します。

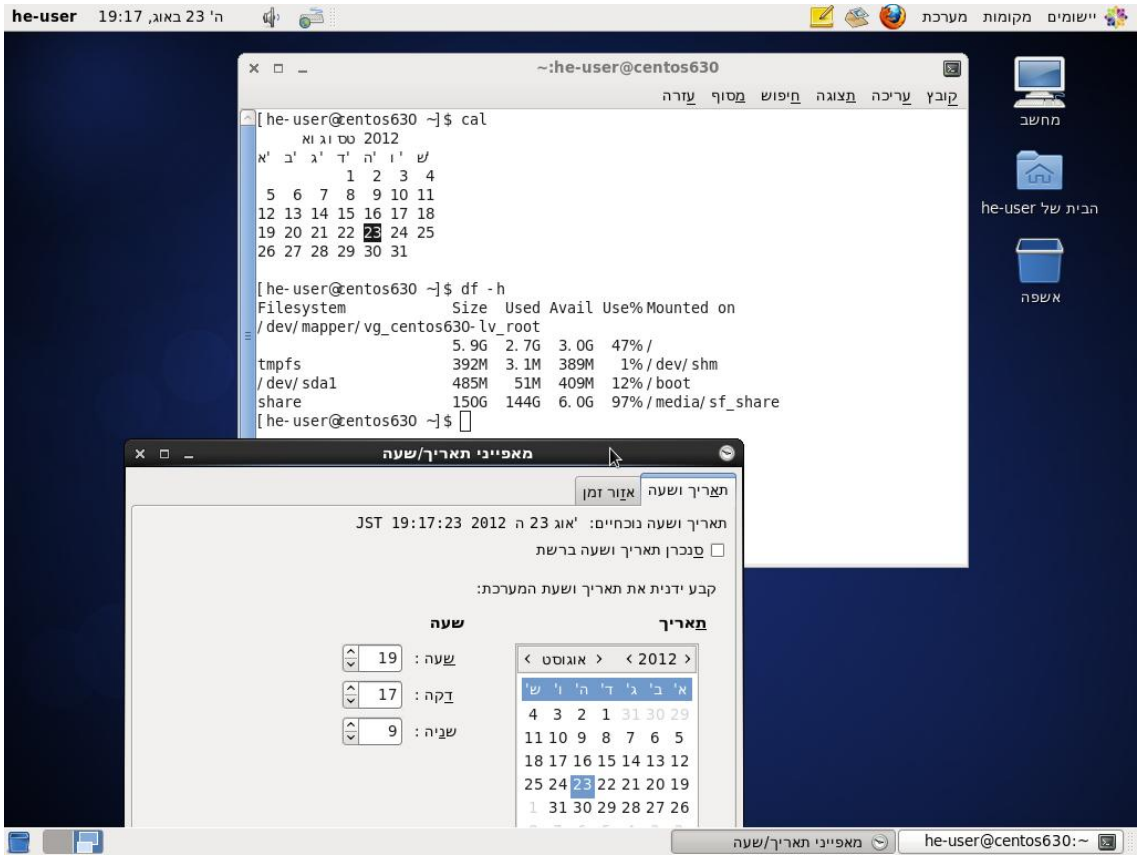
ロケール：ar_AE.utf8

アラビア語は RtoL といって、右から左に文字を書くため、メニューの並びも、アイコンの配置も逆になります。

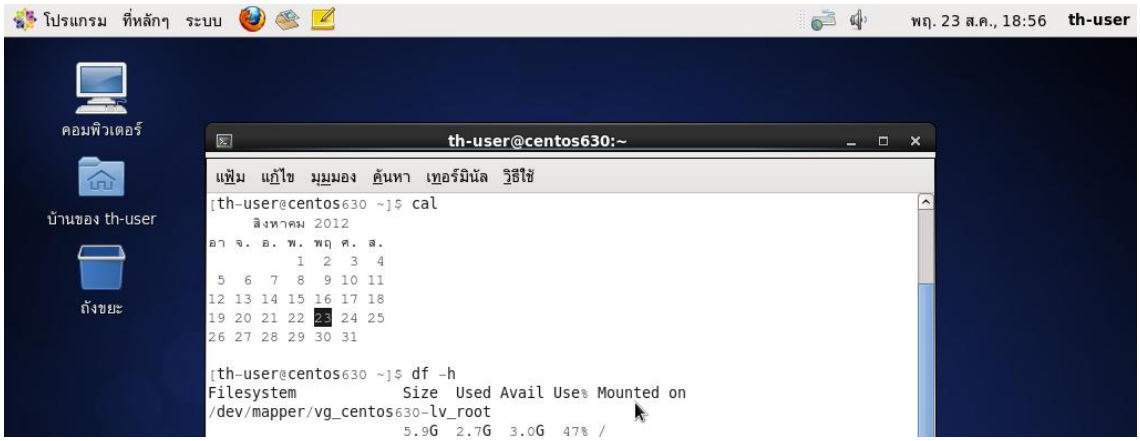


ロケール：he_IL.utf8

ヘブライ語はアラビア語・ペルシャ語同様 RtoL ですが、品詞毎に文字が連なる事はありません。



ロケール:th_TH.utf8
タイは L10N が比較的早い時期に始まっており、1990 年時点の DEC では、日本とタイのみ L10N。
また仏歴も使う場合があります。iPhone ではグレゴリオ暦(Greagrian-2012)、和歴(Japanese-H.24)、仏歴(Buddhist – BE2555)が実装されている。



iPhone の例
【設定アイコン】>一般 からロケール等の設定ができます。

Windows はロケールを変更するには追加ライセンスが必要だったり、英語+1 種類といった制限が課せられる場合が多く手軽に試すのは難しいです。DOS の場合は、chcp(change codepage)でフォントの変更を行う事ができます。
例)コードページ:日本語 932 (Shift-JIS), 米 437

```
C:¥xampp¥htdocs¥LA¥09.ENTER>chcp
現在のコード ページ: 932
C:¥xampp¥htdocs¥LA¥09.ENTER>chcp 437
Active code page: 437

C:¥xampp¥htdocs¥LA¥09.ENTER>dir
 Volume in drive C has no label.
 Volume Serial Number is A898-8E21

 Directory of C:¥xampp¥htdocs¥LA¥09.ENTER
```

メッセージの分類

ここではコンピュータが発するメッセージをいくつかの分類に分けて、紹介します。
特に一般ユーザが見ることのない意味不明なものを中心に、その見極めがしやすいよう留意して解説します。

緊急度による分類

メッセージに緊急度(重用度)を対応づけておけば、利用者も対応がとりやすく間違いを犯す確率を下げる事ができます。多くのシステムでもこの考えは導入されていて、緊急度に応じメッセージの色を変える、ウインドを出すなど表現方法を変で必要に応じ目立たせる取り組みがされています。

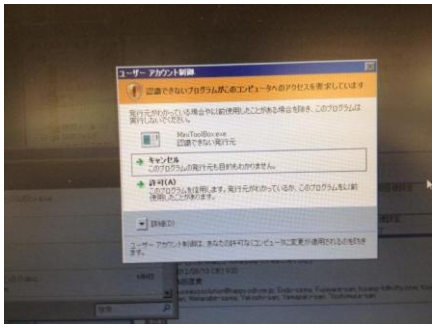
例)Linux のシステムメッセージの分類(syslog)

Emergency	コンピュータを停止しなければならないほどの異常、緊急事態。
Alert	直ぐに何らかの対策を取らなければいけない事態の発生
Critical	重大な障害、事象の発生。Alert につながる危険性。
Error	何らかのトラブル。サービスは停止するがシステムは動作。
Warning	警告。内容によっては無視しても構わない。
Notice	正常だが、重要な内容。放置すると Warning になる可能性。
Information	情報提供のみ。特に対処する必要はない。
Debug	デバッグ用情報。通常は発せられない。

例)PHP の文法チェックにおける分類(php.ini)

ERROR	プログラムの間違い、処理を中止。
WARNING	プログラム内に矛盾があるが、処理続行。
NOTICE	厳密には不備があるものの、PHP 自身が修正して処理続行。
STRICT	古い文法で記述されている。将来使えなくなる可能性がある。

例)Windows のシステム設定に関する確認画面(Dialog box)



Windows Vista 以降、システム全体に影響を及ぼすような操作については、背景を暗くし操作の重大さを訴求するデザインとなっている。
左図はソフトウエア(Oracle Virtual Box)のインストール中にネットワーク設定の変更を行う際に表示されたもの。

基本的に Notice, Warning, Error の順に状況が悪く、OS やプログラミング言語にかかわらず用いられる共通な表現となります。

Emergency はよほどのことで、大抵はコンピュータシステムの範疇を超えたトラブルに用いられます。たとえばシステムが火を噴いたとか、プリンターに人が挟まれたとか…

また Error(間違い)を検出することを Verification といいますが、似た用語に Validation があります。Verification は機械的、杓子定規に内容を確認することで、たとえば入社月日に ABC といった数字以外の入力や、2 月 31 日といったあり得ない日付が設定されていないかを確認する事を指します。Validation は文脈、意味の解釈に近く、妥当性検証ということもあります。例えば入社月日に 1642 年 12 月 25 日[2]といった「日付」としては正しくても、会社が設立するよりも前であるという矛盾をチェックする事を指します。英語圏ではこのように明確な使い分けをする事が多く、コンピュータ業界でもその文化が定着しています。

[2] アイザック・ニュートンの誕生日

保存期間による分類

コンピュータメッセージには、都度状況報告のために表示されるものと、恒久的に記録されるものがあります。利用者に対してエラーを通知するものは都度表示が多く、システム全体に影響するものや許認可の証左に関わるものは記録されます。

保存されるメッセージ

恒久的に記録されるメッセージはログ(log:航海日誌から)と呼ばれ、システム運用で常時内容が確認されます。多くはファイルに記録され長期間保存されます。金融、特に証券関係では「誰がいつメールをやりとりしたか」といった通信記録もインサイダー取引防止の観点から重要視されています。このようなログは監査対象となっており、改竄されないよう厳重に防御されています。

例)メールログ (linux)

```
Sep 18 10:45:27 cent580 postfix/pickup[2445]: 900E71004FA: uid=500 from=<student>
Sep 18 10:45:27 cent580 postfix/cleanup[2591]: 900E71004FA: message-id= <201209180
14527.900E71004FA@smtp.t14106.la.net>
Sep 18 10:45:27 cent580 postfix/qmgr[2446]: 900E71004FA: from=<student@t141006.la.
net>, size=347, nrcpt=1 (queue active)
Sep 18 10:45:27 cent580 postfix/local[2593]: 900E71004FA: to=<student@t141006.la.
net>, orig_to=<root>, relay=local, delay=0.44, delays=0.19/0.15/0/0.09, dsn=2.0.0,
status=sent (delivered to command: /usr/bin/procmail)
Sep 18 10:45:27 cent580 postfix/qmgr[2446]: 900E71004FA: removed
```

大抵の場合はその量が膨大になりますから、定期的な削除(古いものをバックアップし、削除)が必要となり運用に手間がかかります。特に重要な情報を記録したログファイルが容量いっぱいで書き込めなくなると、システム全体が停止するといった障害に発展する場合もあります。

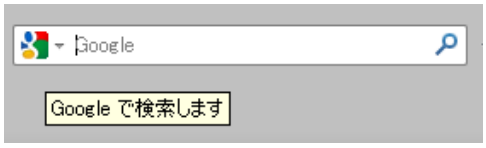
利用者が確認するまで留まるメッセージ

利用者の確認、または指示があるまで留まるタイプのものです。GUI の世界では一般的で、ダイアログボックス(Dialog box)と呼ばれます。

Dialog box の中でも、利用者が回答するまで処理を中断するものを nodal dialog と呼びます。
左図のように、答えない限りアプリが停止するものを「Application modal」、先のシステム設定のようにOS 自体が停止するものを「System modal」と呼びます。modal は「様相」という意味があり、とくにこの dialog については、必然性や可能性といった概念を指し示す意味に使われています[3]。難しいですね。

都度表示されるメッセージ

ユーザに対し状況を報告しますが、特に確認することなく一定時間が経つと消えてしまうメッセージです。または、簡単な利用方法などが表示されるタイプのものもあります。



最近よく用いられるのは、Balloon Window(or Help)と呼ばれるもので、入力項目にカーソルを位置付け、しばらくすると「使い方が分からないのかな?」と気をまわして聞いてきてくれるタイプのものです。

「少し時間をおくと表示される」という点がミソで、随時表示されると利用者からはうっとうしいですし、システムにもとても負荷がかかり、全体として処理効率が落ちてしまいます。

---

[3] Modal Logic - 様相論理学

表現方法による分類

コンピュータが発する最も原始的なメッセージは、「戻り値」や「終了コード」と呼ばれる数値です。範囲も多くはなく、これを読み取って分かる意味は数種類から多くても十数種類程度です。

単純な数字

例） Windows (MS-DOS)

```
C:¥ >dir x
 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 A898-8E21 です

 C:¥ のディレクトリ

ファイルが見つかりません

C:¥ >echo %ERRORLEVEL%
1
```

例えば正常終了が0で、それ以外はコード表に照らし合わせて原因を追及します。物によっては二進数になっていて、各ビットに意味を持たせている場合もあります。

Web やメールサーバは 3 ケタの数字で状態を表し、百位で凡その意味が決められています。

コード	Web サーバ	メールサーバ
1xx	情報提供(Information)	処理開始前の情報提供
2xx	正常に処理	正常に処理
3xx	代理サーバの回答(redirect)	処理の中間報告
4xx	ブラウザ側のエラー	再実行可能なエラー
5xx	サーバ側のエラー	サーバ側の回復不可能なエラー

表意記号

もう少し意味を持たせた場合は、英数字を組合せた表意記号を用います。この記号をもとにコードブックという小冊子を引くと、そこに原因や対策が記述されているものもあります。昔の大型計算機ではよく用いられ、OS、プログラミング言語などソフトウエア毎に整理されていました。

例）汎用機のメッセージ例。IEFxxI が記号

```
IEF212I JOB001 STEP1 DSN0A - DATA SET NOT FOUND
IEF253I JOB001 STEP2 DSN2B - DUPLICATE NAME ON DIRECT ACCESS VOLUME
```

この例は IBM をはじめとする汎用機で使われていたもので、種々の用語すら分かりづらいと思います。

DATA SET	今でいうファイル
DIRECT ACCESS VOLUME	同じくハードディスク

エラーメッセージ

より利用者にわかりやすい言葉で状況を表す方法です。といっても、一般ユーザ向けではないのでわかりづらいと思います。特に英語で表示されるとお手上げの人も多いのではないでしょうか。
次に、このエラーメッセージ、しかも英文に注目して解説します。

メッセージ例

いろいろなシステムで表示されるエラーメッセージの中から、よく目にするキーワードを抽出しました。またキーワードに着目して、その謂れや原因も解説します。一般の英語表現ではあまり用いない部分については、それも合わせて解説します。

dump（ダンプ）

Linux では、プログラムの異常終了時 Core dumped や、Kernel dump など、Windows では、突然画面が真っ青になった場合に見受けられます。

dump は「どさっ、と落とす」の意味で、ダンプカーは砂利を「どさっと落とす」ところから名づけられました。
コンピュータの場合は砂利ではなく、そのプログラムが使っていたメモリの内容全て（または一部）を、ファイルに書き出したという意味になります。

「さっきまで使っていた DVD が急にどこかへ消えた」「指定されたメモリ領域にアクセスできない」といった、プログラム単体では解決できそうにない致命的なエラーが起こったとき、その状況を記録するための行為です。

Windows ではハードウエアの故障や、ドライバの設定ミスが考えられます。Linux では、Windows ほど致命的ではなく、入力データが存在しない、関連するプロセスの異常などが考えられます。
OS に関係なく、dump したデータを解析するためには、プログラムのソースコード（設計図）が必要となります。
ちなみに Linux で見受けられる Core dump の Core は、昔使われていた主記憶の部品「磁気コアメモリ」が由来となっています。Windows（図）でいう「physical memory(物理的なメモリ)」と同じ意味です。

syntax（シンタックス）

プログラミングやシステム設定で定義ファイルを修正する人にはおなじみの単語です。Syntax は「文法」の意味で、指定した内容に明らかな誤りがある場合に発せられます。
相手は機械ですので、わかりきった内容であってもルール通りの記載でなければ、突き返されます。

例 1）Java のプログラミング

```
----------
1. ERROR in hello.java (at line 3)
   System.out,println("Hello world!");
              ^
Syntax error on token ",", . expected
----------
```

3 行目の System~の行にある、out の後ろのカンマ(,)が文法的に間違いです。
もしかしてドット(.)ではないですか？

例 2）BIND の定義ファイルの誤り

```
  1  // Sample named.conf
  2  //
  3  options
  4  {
  5          directory "/var/named";
  6          dnssec-enable   yes
  7  };
```

左の設定ファイルの内容で、このプログラム(BIND)を起動させると、次のようなメッセージが表示されます。
Error in named configuration:
/etc/named.conf:8: missing ';' before '}'
8 行目の “:” より前で、「;」 が抜けている。

どちらも「そこまで分かっているのなら、直しておいてよ」と思うのですが、機械が勝手に判断する事の方が問題なので、まさに「機械的な反応」をします。まぁ地方のお役所も似たり寄ったりですが…。

Not enough / Insufficient

Not enough も Insufficient も「不足」を意味します。多くの場合はコンピュータ資源が不足して、動作に影響がある場合に表示されます。

```
Not enough space                (Solaris)
```

この場合、space はディスク容量を意味し、空き容量の不足となっている場合に表示されます。

```
Insufficient memory
```

この場合は、そのままの意味で、メモリ不足です。Insufficient は単純なコンピュータ資源だけでなく、権限の不足といったより複雑な場合にも用いられます。

```
ORA-01031: insufficient privileges.
```

このように、insufficient の方が解決に手間取りそうな場合が多い気がします…。
メモリの扱いは、初心者が思うよりも複雑で、システムに負荷を強いる場合があります。プログラムが起動して終了するまで、最も変動するのがメモリでしょう。大きな地図を必要な部分だけ切り取って表示をする、MAP アプリなどは、必要に応じて表示していない部分も取り込みます。表示・非表示にかかわらず必要となるのがメモリです。
メモリをよく使うアプリは、終了する際にちゃんと表示していない部分のメモリも後始末(解放)する必要があります。このように、気を付けて「良い仕事」をしないとメモリ不足になる場合があります。
アプリが終了しても、解放されなかった領域は使用中と見なされ、システム全体の空き容量がジワジワと減っていく場合があります。一番簡単な方法はシステムの再起動(リブート)ですが、近年では 24 時間 365 日稼働しないといけないシステムもあります。
この様にアプリが終了しても、メモリを消費したままになることを memory leak と呼び、多くのプログラマを悩ましています。

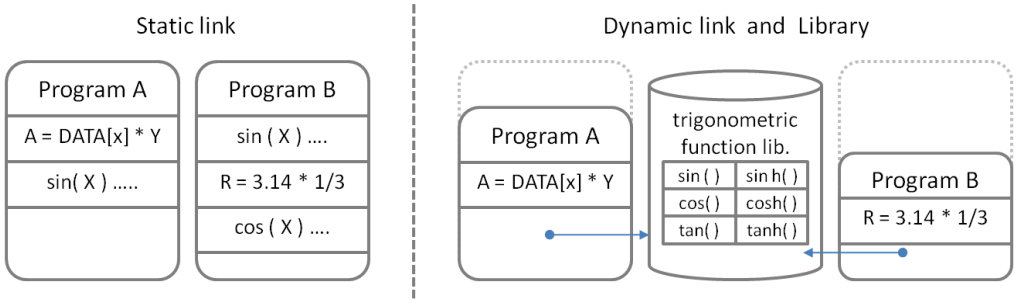
undefined

「定義がない」という意味ですが、そもそも「define 定義」とう言葉はプログラミングで使われています。データを格納する変数や、日付順に並び替える機能、そういったものに名前を付け、どういった内容かを明記することを「定義」といいます。
この定義は、プログラムを作る際にチェックされるため、本来利用者が見る事はありません。

例)プログラムを作成したが、「関数 sin()が定義されていない」とういエラーが発生。

```
/tmp/ccwRFaTf.o: In function `main':
a.c:(.text+0x1b): undefined reference to `sin'
collect2: ld returned 1 exit status
```

プログラムから汎用的に使われる機能は、ライブラリというかたまりで管理されます。ライブラリにする事で、バージョンの一元管理とディスク容量の節約が可能となります。



この用に複数のプログラムで共有されるライブラリを、Shared object(共有オブジェクト)や、Dynamic Link Library (DLL:動的配置ライブラリ)と呼びます。
プログラムはインストールされているのに、ライブラリのインストールを忘れた場合などに、undefined が発生します。

invoke

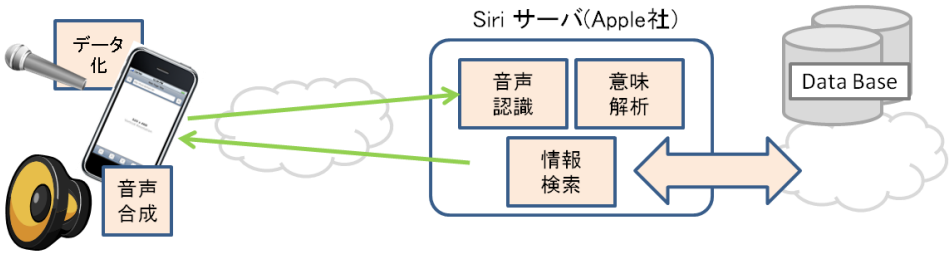
呼び出すという意味ですが、call に比べて非常に馴染みのない単語です。たぶん大学受験でも出ないのではないかと思います。辞書を引くと「加護などを懇願する」「拒否権を発動する」「精霊を召喚する」など、普段使わない文例ばかりです。

```
Failed to Invoke [function name]
```

invoke は Java などで、特定の機能を呼び出す場合に用いられます。従来のプログラミング言語では、機能を呼ぶことを call と呼称していました。特に使い方を間違うと、

```
Illegal function call function name
```

といったメッセージが普通でした。
Java, JavaScript, PHP などなると、プログラムがネットワークを介して別のプログラムに協力を依頼するといったことが多くなりました。特にクラウド・コンピューティングに代表されるように、携帯デバイスから、大型コンピュータを呼び出して処理をするといった所が当たり前になりました。まさにスマフォから、魔神を「召喚」している様に見えるので、この言葉が今後定着するかもしれません。



Permission / Privilege

permission(パーミッション)は許可・許諾、privilege(プリビレッジ)は特権を意味します。どちらもファイルを削除するとか、データを修正するといった特定の操作に対して、何らかの制限があり「あなたは、その操作を行う権限がありません」といった時に使われます。

Permission は操作を行おうとしている対象(ファイルや、データ)に付与された属性に「持ち主以外、変更しちゃダメ」、「管理部門以外、みちゃだめ」といった許可属性が付与されている場合に多く用いられます。

Privilege は操作その物に対して権限があるかどうかを意味し、「あなたはデータを見る事はできますが、変更する権限はありません」といった時に用いられます。Permission が物に対して付与された許可であるのに対し、Privilege は操作する人に与えられた権限・特権です。

```
[student@cent580 ~]$ mv /etc/hosts /etc/hosts.bkup
mv: cannot move `/etc/hosts' to `/etc/hosts.bkup': Permission denied
```

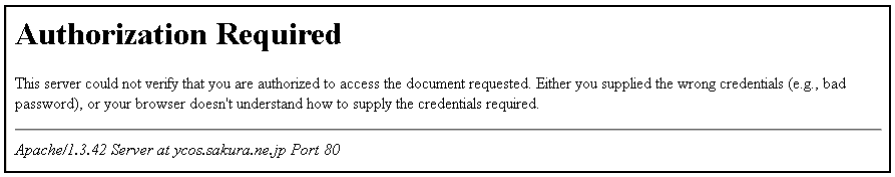
これは Linux でファイル名を変更しようとした例ですが、Permission denied (許可、否認) = 許可がありません、となっています。実際にこのファイルは特定のユーザ以外操作が認められない設定になっているので、特定のユーザに依頼して名前を変更してもらうか、私でも変更できるようファイルの許可属性を変更してもらう必要があります。

```
ORA-01045: user ユーザ名 lacks CREATE SESSION privilege; logon denied
```

これは Oracle DB に Web サービスが接続する際のエラーです。Web サービス用に作った「ユーザ」が、ネットワークを経由して DB を操作する権限(CREATE SESSION)がないと言っています。
情報へのアクセスについて、「情報その物」ではなく「操作する人」に着目したのはリレーショナル・データベースが最初に広めたため、今でも Privilege という用語はデータベース関係で多く見られます。

authorized / authentication

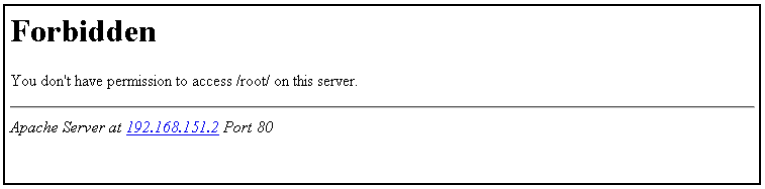
authentication は認証という意味ですが、一般にはコンピュータが利用者を特定する、本人確認全般を指します。一番簡単な認証方法はユーザ名とパスワードの組み合わせです。他にも乱数表を用いる方法 one time password や、彩光・指紋・静脈・顔といった固有のパターンを使った biometrics もあります。Authorized は、認証により本人が確認された状態を示します。



また Certification は証明、認定で、第三者によって身分を保障してもらう事を意味します。例えばECサイトでの買い物で、そのサイトが正しい商取引相手である事は、CA(Certification Agent)という、第三者機関によって証明されます。
技術資格の CCNA(Cisco Certified Network Associate), LPIC(Linux Professional Institute Certified)などもベンダーや中立 NPO などが、その人の技量を証明してくれています。

Forbidden

Forbidden は禁じされた、ご禁制のという意味があり非常に形式ばった言い方です。the forbidden fruit は「禁断の果実」です。アダムとイブが食べてしまい、エデンの園を追放されたアレです。
従来は、このようなアクセスエラーでは permission denied や、Not enough privilege が一般的でしたが、Web の世界で広がりました。

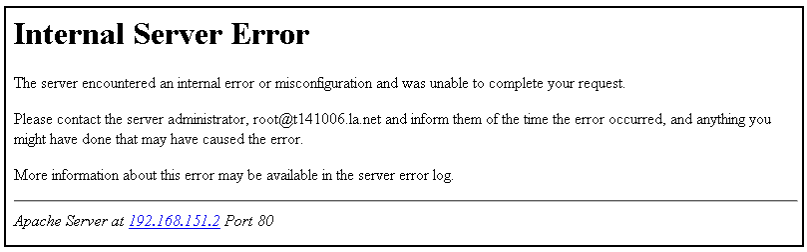


従来のシステムであれば、全く知らない人物からのアクセスは考えられませんでした。たとえ特定のファイルにアクセスできないとしても、少なくともシステムの利用権はある人でした。会社でいえば、人事情報にアクセスできなくても、オフィスに出入りできるユーザに限定されていました。
Web の登場で全く見ず知らずの人が、システムにアクセスしてくるようになると、従来のシステムとは異なり、情報漏洩に気を使うようになりました。これが、より強い言い回しである「禁断」を使った謂れでしょう。
ただ、禁断の実だから食べたくもなるわけですし、見るな！といわれると見たくなるのが人情…。
そういった気持を表しているのかもしれません。

Web の世界では他にも従来のコンピュータシステムと異なる言い回しが使われます。たとえばホームページが一時的にサービスを停止している場合、under construction（工事中）と表記するサイトが多く登場しました。最近では、サーバの無停止度合いが向上して非常に見かけなくなりましたが…

Internal Server Error

これも Web Server で特徴的なメッセージです。



普通に考えれば、システムの不具合なんだから「サーバの中で起こっている」だろう…。

付録

コード表

ASCII (7bit)

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
20		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
30	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
40	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
50	P	Q	R	S	T	U	V	W	X	Y	Z	[	¥	]	^	_
60	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
70	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	

なお ASCII では 0(00)～31(1F)までは制御文字が割り当てられている。7F は末梢でデータの削除を意味し、1 文字後退(Back Space)とは区別されている。

EBCDIC (Extended Binary Corded Decimal Interchange Code)

IBM が独自に採用した文字コード。多言語に対応するよう複数の表を持ち、切り替えて使用することができる。各表にはコードページと呼ばれる番号が割り振られている。ASCII と異なりアルファベットや数字が非連続の配置となっているが、これは開発当時のハードウエアの制限によるものと言われている。

行＼列	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
40			。	「	」	、	・	ヲ	ァ	ィ	¢	.	<	(	+	|
50	&	ゥ	ェ	ォ	ャ	ュ	ョ	ッ	-	ア	!	$	*	)	;	¬
60	-	/	イ	ウ	エ	オ	カ	キ	ク	ケ	|	,	%	_	>	?
70	コ	サ	シ	ス	セ	ソ	タ	チ	ツ	`	:	#	@	'	=	"
80	テ	a	b	c	d	e	f	g	h	i	ト	ナ	ニ	ヌ	ネ	ノ
90	ハ	j	k	l	m	n	o	p	q	r	ヒ	フ	ヘ	ホ	マ	ミ
A0	ム	~	s	t	u	v	w	x	y	z	メ	モ	ヤ	ユ	ヨ	ラ
B0	リ	ル	レ	ロ	ワ	ン	゛	゜								
C0	{	A	B	C	D	E	F	G	H	I						
D0	}	J	K	L	M	N	O	P	Q	R						
E0	¥		S	T	U	V	W	X	Y	Z						
F0	0	1	2	3	4	5	6	7	8	9						