コラム：テキスト処理のための言語

本書ではプログラム言語として Python を主に使用していますが, Python だけが唯一の言語というわけではありません. Python の前に一世を風靡した Perl [30]は, 非常にテキスト処理に長けていました. 作者の Larry Wall はバークレー校で言語学を学んでいたため, Perl の構文には`$@&`による変数の「品詞」, `$_`による「痕跡」など, 言語学の要素が多く取り入れられ, その機能の一部は Python にも継承されています.

また, テキストを扱うのを得意とする `awk` や `sed` といった言語が, Linux や MacOS のような Unix には古くから標準的に含まれており, 利用できます.[*53] 44 ページの脚注でも使用した `awk` は, テキストを 1 行読むごとにフィールドを空白で自動的に分割して`$1`,`$2`,…という名前をつけてくれますので, たとえばテキストの `Foo` で始まる各行の 2 個目のフィールドの総和を求めたければ,

```
% awk '/^Foo/{ s+=$2 };END{ print s }' input.txt
```

のように簡潔に書くことができ, 通常は Excel で行うような計算を簡単に行うことができます. `awk` は他にも exp, log, sin のような算術演算や `substr` のような文字列演算, `for` 文による繰り返しなども備えており, 簡単なデータ解析にもたいへん有用です.

`sed` は stream editor の略で, 1 行しか画面出力を持たないエディタ `ed` [*54] の拡張ですが, それゆえに簡潔なコマンド体系を持っています. たとえば

```
% sed '1,5d;s/foo/bar/g' input.txt
```

とすれば, 1 行目から 5 行目を削除 (`d`) した上で, `foo` をすべて (`g`)`bar` に置換 (`s`) して出力します.

*53 これは, 歴史的に Unix がテキスト処理をその目的の一つとして開発されたためです[].
*54 `ed` については, 『The UNIX Super Text』[31]などを参照してください.

また

```
% sed G input.txt
```

とすれば, 1 行おきに空白を入れた「ダブルスペース」のテキストを簡単に作ることができます. パターンスペース, ホールドスペースといった内部の記憶領域をうまく使うと, さらに高度な処理も可能で, 何と sed で書かれたテトリス [*55] やチェス [*56] すら存在します.

`sed` や `awk` については『sed&awk プログラミング』[32]『プログラミング言語 awk』[33] といった教科書があるほか, 「awk の簡単な使い方」[*57]「sed 教室」[*58]「sed は日暮れて」[*59] といったフリーのチュートリアルがあります. こうした言語を適宜使いこなすことで, テキストを計算機でより自由に扱えるようになるでしょう.

なお, 本書を書く際に用いた LaTeX (TeX) も, テキストを処理するプログラミング言語の一種といえます. チューリング完全, すなわち Python のような言語と同じ表現能力を持っているため, TeX で書かれた BASIC インタプリタ BASIX[*60] [34]など, 驚くべきプログラムも存在しています.

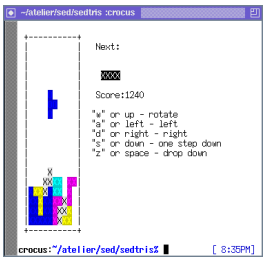

図 1.22: 文字端末で sed によるテトリスをプレイしている様子.

*55　https://github.com/uuner/sedtris
*56　https://github.com/bolknote/SedChess
*57　「awk の簡単な使い方」http://chasen.org/~daiti-m/etc/awk/ に転載.
*58　「sed 教室」https://www.gcd.org/sengoku/sedlec/
*59　「sed は日暮れて」https://chimimo.tumblr.com/post/8995558289/sed1
*60　https://www.ctan.org/tex-archive/macros/generic/basix