本パッケージは「ページスタイル」の機構を SATYSFI に提供します．「ページスタイル」とは，ヘッダやフッタを意味します．クラスファイル作成者向けのパッケージです．本パッケージは以下のパッケージに依存しています．

- `satysfi-fss`：フォント管理を行うパッケージ．
- `satysfi-pagenumber`：ページ番号の管理を行うパッケージ．

クラスファイルに本パッケージを組み込むには次の手順を踏みます．

(1) ヘッダなどに出力する見出し文字列などを「マーク」に登録する．通常 `+section` などの見出し命令内で行う．
(2) ページスタイルを作成し，登録する．
(3) `page-break` の第三引数（など）に本パッケージからの出力を渡す．

順番に説明します．

1. マーク

例えば `+section{はじめに}` という見出しがある場合，ヘッダに「はじめに」と出力させたいことはよくあります．マークはこのような機能を作るための支援を行います．具体的には，`+section{はじめに}` において「マーク」に「はじめに」を登録しておき，ヘッダを出力する際にこれを参照するということができるようになります．

マークを使うために `mark` モジュールを読んでおきます．

```
@require: pagestyle/mark
```

マークの登録は `Mark.set-mark` を使います．

```
let-block ctx +section ?:label title inner =
  % 0 番目のマークに title を設定する
  % inline-nil と等価な inline-boxes がかえる
  % 実際には「1 節」のような文字列も同時に登録しておくことが多いだろう．
  let mark-b = Mark.set-mark 0 title
  ....
  % きちんと Mark.set-mark の戻り値を埋め込んでおく．
  mark-b ++ main-b
```

Mark.set-mark は次のように定義されています.

```
set-mark: int -> inline-text -> inline-boxes
```

Mark.set-mark index txt の形で使い，index 番目のマークに txt を登録します．戻り値は inline-nil と等価な inline-boxes です．この戻り値を必ず本文に埋め込んでおきます.

ヘッダなどに出力する際には set-mark の第一引数に渡した整数値を使います．別名をつけておくと便利でしょう.

```
let section-index = 0
```

読み出すには次のような命令を使います．ただしこれらの命令が機能するのは各ページのページ数が確定した後です．なお，本パッケージのページスタイルの機能を使う場合はこの登録した値を読み出す必要はありません.

```
% index 番目の，pn ページにおける最初のマークを取得する
let first-mark = Mark.get-first-mark index pn
% index 番目の，pn ページにおける最後のマークを取得する
let last-mark = Mark.get-last-mark index pn
```

これらで得られるマークは，pn ページ目で登録されたものとは限りません．仮に pn ページ目ではマークが登録されていない場合には，それ以前に登録されたものが得られます.

2. ページスタイルの作成

ページスタイルは以下のように定義されています.

```
type page-style = (|
  odd-header : context -> int -> int -> string -> block-boxes;
  even-header : context -> int -> int -> string -> block-boxes;
  odd-footer : context -> int -> int -> string -> block-boxes;
  even-footer : context -> int -> int -> string -> block-boxes;
|)
```

各関数の第二引数は真のページ数（文書の最初から一ずつ増えていくもの），第三引数はみかけのページ数，第四引数はみかけのページ数を文字列で表したものです．（これらのペー

ジ数の違いについては PageNumber パッケージの説明書をご覧ください．）これを直接設定してもよいのですが，次の page-style-scheme を使っても作成することができます．まずはページスタイルの機能を使うために PageStyle モジュールを読み込んでおきます．

```
@require: pagestyle/pagestyle
```

次のようにしてページスタイルを作成することができます．

```
let my-page-style = PageStyle.page-style-scheme (|
  nombre = [% ノンブルの指定
    (|
      % 場所はフッタの真ん中
      position = PageStyleBottomCenter;
      % ノンブルの出力．ページ数をそのまま出力する．
      nombre = (fun pbinfo ps -> embed-string (ps));
      % フォント指定. satysfi-fss での指定をする．
      font = [with-font-size (fun l -> l *' 0.8)];
    |);
  ];
  running-head = [% 柱の指定
    (|
      % 柱の場所はヘッダの中心
      position = PageStyleTopCenter
      % 奇数ページには +subsection の見出しを出力
      odd = PageStyleFirstMark(subsection-index);
      % 偶数ページには +section 見出しを出力
      even = PageStyleBotMark(section-index);
      font = [with-font-size (fun l -> l *' 0.8)];
    |)
  ];
|)
```

ただし section-index や subsection-index には適切な整数値が束縛されていて，+section や +subsection において対応するマークが設定されているとします．

position には PageStyleBottomCenter，PageStyleBottomLeft，PageStyleBottomRight，

PageStyleTopCenter, PageStyleTopLeft, PageStyleTopRight の六つを設定できます.

running-head における odd や even には以下のどれかを指定します.

- PageStyleFirstMark(n) n 番目のそのページ最初のマークを出力します.
- PageStyleBotMark(n) n 番目のそのページ最後のマークを出力します.
- PageStyleTopMark(n) n 番目のそのページ頭でのマークを出力します.
- PageStyleFormat(f) f で定まるヘッダやフッタを直接指定します. f は型 int -> string -> inline-text を持ち, f pn pstr, ただし pn は現在の見た目のページ数, pstr は見た目のページ数を文字列にしたものとして呼び出されます.

さらに, 第一引数にオプションで次の引数を与えることができます.

```
(|
  header : context -> block-boxes -> block-boxes;
  footer : context -> block-boxes -> block-boxes;
|)
```

ヘッダおよびフッタは出力直前にこの関数に代入され, その結果を最終的に出力します. 例えばヘッダに下線を引くには次のようにします.

```
let underline ctx b =
  let pads = (0pt, 0pt, 0pt, 2pt) in
  let deco = (fun (x,y) w h d -> [
    stroke 0.5pt Color.black (Gr.line (x,y -' d) (x +' w,y -' d));
  ]) in
  let decos = (deco,deco,deco,deco) in
  block-frame-breakable ctx pads decos (fun c -> b)
let my-page-style = PageStyle.page-style-scheme ?:(|
    header = underline;
    footer = (fun _ b -> b);
  |)% 第二引数は上で説明したもの
```

ページスタイルの登録は PageStyle.set-page-style: page-style -> unit で行います.

```
set-page-style my-page-style
```

本文中で変更するには set-page-style-inline を使います.

```
let ps-b = set-page-style-inline my-page-style in
...
% 戻り値は inline-boxes，本文に埋め込む.
main-b ++ ps-b
```

これのユーザ用コマンドである \set-page-style もあります．文書内で以下のように呼び出すことでページスタイルが変更されます.

```
+p{
  ...
  % 現在のページスタイルがこれ以降変更される.
  \PageStyle.set-page-style(new-ps);
  ...
}
```

set-this-page-style-inline を使うと，現在のページのページスタイルを一時的に変更できます．次のページに行くと元に戻ります．ユーザ用コマンドは \set-this-page-style です.

```
+p{
  ...
  % このページのページスタイルだけ変更される.
  \PageStyle.set-this-page-style(new-ps);
  ...
}
```

3. ページスタイルの出力

PageStyle.header と PageStyle.footer を使うと，現在登録されているページスタイルに従った出力を得ることができます．主にクラスファイル内で定義された document 関数

内で呼び出されることになるかと思います．次の使用例を見ればその使い方がわかるでしょう．

```
...
let pagestyle pbinfo =
  let ctx = ... in % context を取得
  let (xh,yh) = ... in % header の位置を取得
  let (xf,yf) = ... in % footer の位置を取得
  (|
    % 引数の `true' はヘッダとフッタがページ数の
    % 偶奇に応じて変わることを意味する.
    % false にすると常に odd と見なされる.
    header-content = PageStyle.header true pbinfo;
    header-origin = (xh,yh);
    footer-content = PageStyle.footer true pbinfo;
    footer-origin = (xf,yf);
  |)
in
...
let paper-size = ... in % 紙サイズを設定
let page-layout pbinfo = ... in % 本文位置を設定
let main-b = ... in % 本文を取得
page-break
  paper-size
  page-layout
  pagestyle % ここに上のページスタイルを与える
  main-b
```