- いい運用です！そのままでも動きますが、安定性・再現性・保守性をぐっと上げるために、以下を直す／足すのがおすすめです。要点→すぐ使える修正例の順でまとめます。

改善ポイント（重要度順）

---

a. **minted を使うなら常に `-shell-escape` ＋ `latexmk` に統一**
   - 今の分岐：SimpleDarkBlue は `lualatex`、metropolis は `latexmk -shell-escape`。
   - でもテンプレに `\usepackage{minted}` が入っているので、**SimpleDarkBlue 側も `-shell-escape` 必須**。
   - さらにページ数収束や依存解決は `latexmk` が自動でやってくれるので、**両テーマとも latexmk に統一**が◎。

b. **ビルド用ディレクトリを分離（中間物を掃除しやすく）**
   - `build/`（または一時ディレクトリ）でコンパイル → 収束後の `main.pdf` だけをアプリ側へコピー。
   - `latexmk` の `-outdir`（+ `-auxdir`）を指定すると吉。

c. **サブプロセスを堅牢化（タイムアウト・要約ログ・明確な失敗）**
   - `timeout=`、`-file-line-error -halt-on-error` を常に付与。
   - 失敗時は `!` 行や `error` を含む行を抽出して最後に出す。

d. **パスを LaTeX 向けに正規化**
   - `@@sdir@@` に OS の `\` が混ざると LaTeX が誤解する。**置換前に `/` に統一**。
   - 可能なら日本語・結合文字を含むパスは避ける。

e. **テンプレ終端の扱い**
   - いま `\end{document}` を必ず足していますが、将来 `content.tex` に入っていると**二重**になります。
   - 末尾がすでに `\end{document}` なら追加しないガードを。

f. **未使用オプションの活用**

- `--page`（範囲指定）をビルド後の PDF トリミングに反映（後述の `pypdf` 例）。

g. **重複や危険箇所の整理（テンプレ側）**

- `xcolor` の二重読み込みを削除（`[table,xcdraw]{xcolor}` のみ）。
- `hyperref` は**最後**に読み込む。
- `footline` は**一つ**に統一（同ファイル内で2回設定されている）。
- `minted` を使うフレームは `[fragile]` を忘れずに（テンプレ／運用ルール）。

すぐ使える修正版（抜粋）

---

1) Python：堅牢ビルド & 共通 latexmk

```
from pathlib import Path
import slideinfo
import subprocess
import argparse
import re
import shutil
import sys
import os
import textwrap

def parse_page_range(range_str: str) -> tuple[int, int]:
    if not range_str:
        return -1, -1
    if "-" in range_str:
        start, end = map(int, range_str.split("-"))
        start = max(1, start)
```

```
        if start > end:
            end = start
        return start, end
    else:
        page = int(range_str)
        return (1, 1) if page == 0 else (page, page)

def find_frame_positions(s):
    pattern = r'(\\begin{frame}.*?\\end{frame})(\n|$)'
    return [(m.start(), m.end()) for m in
re.finditer(pattern, s, flags=re.DOTALL)]

def themechk(first_line: str) -> str:
    m = re.search(r"@@@--\((.*?)\)--@@@", first_line)
    if not m:
        return "SimpleDarkBlue"
    val = m.group(1)
    if val in {"metropolis", "SimpleDarkBlue"}:
        return val
    print(f"テーマの値が不正です: {val}", file=sys.stderr)
    sys.exit(1)

# ---- CLI ----
p = argparse.ArgumentParser()
p.add_argument("items", nargs=2)
p.add_argument("--page", "-p", default="")
p.add_argument("--ho", action="store_true")
p.add_argument("--tech", action="store_true")
args = p.parse_args()

subj_code, tdir_name = args.items
tagdir = slideinfo.slidedir(subj_code, tdir_name)
if not tagdir:
    sys.exit(1)

# 入出力パス
root = Path(__file__).parent
app_dir = root.parent / tagdir
content_path = app_dir / "content.tex"
```

```
build_dir = root / "build"  # 専用ビルドディレクトリ
build_dir.mkdir(exist_ok=True)

# content 読込み
text2 = content_path.read_text(encoding="utf-8")
ctheme = themechk(text2.splitlines()[0] if text2 else "")
print(f"対象ディレクトリ: {tagdir}")
print(f"beamerテーマ: {ctheme}")

# テンプレ読込み
templ_map = {"SimpleDarkBlue": "main_template_org.txt",
             "metropolis":   "metro_template_org.txt"}
templ = (root / "templates" /
templ_map[ctheme]).read_text(encoding="utf-8")

# 置換（LaTeX向けにパスを / に）
sdir_tex = str(tagdir).replace("\\", "/")
stitle = f"{tdir_name} {slideinfo.slidetitle(subj_code,
tdir_name)}"
text = (templ
        .replace("@@sdir@@", sdir_tex)
        .replace("@@stitle@@", stitle))

# フラグ（pause/teacher）
text = text.replace("%@@pausemode@@", r"\mypausemodefalse"
if args.ho else r"\mypausemodetrue")
text = text.replace("%@@teachermode@@", r"\teachermodetrue"
if args.tech else r"\teachermodefalse")

# main.tex を build/ に生成（テンプレ + 本文）
main_tex = build_dir / "main.tex"
body = text2.rstrip()
out = [text, "", body]
if not body.endswith(r"\end{document}"):
    out.append(r"\end{document}")
main_tex.write_text("\n".join(out) + "\n", encoding="utf-8")

# latexmk（両テーマ共通：minted 想定で -shell-escape）
cmd = [
```

```
    "latexmk", "-lualatex", "-shell-escape",
    "-interaction=nonstopmode", "-file-line-error",
    "-halt-on-error",
    "-outdir=" + str(build_dir),
    str(main_tex)
]
print("RUN:", " ".join(cmd))

try:
    res = subprocess.run(
        cmd, cwd=build_dir, capture_output=True, text=True,
timeout=180
    )
except subprocess.TimeoutExpired:
    print("❌ タイムアウトしました（180秒）", file=sys.stderr)
    sys.exit(1)

if res.returncode != 0:
    # エラーログ要約
    lines = []
    for ln in (res.stdout.splitlines() +
res.stderr.splitlines()):
        if ln.startswith("! ") or "error" in ln.lower():
            lines.append(ln)
    tail = "\n".join(lines[-60:]) or res.stdout[-3000:]
    print("❌ LaTeX コンパイル失敗\n--- LOG ---\n" + tail,
file=sys.stderr)
    sys.exit(1)
else:
    print("✅ LaTeX コンパイル成功")

pdf_path = build_dir / "main.pdf"
if not pdf_path.exists():
    print("❌ main.pdf が見つかりません", file=sys.stderr)
    sys.exit(1)

# 出力ファイル名
title = slideinfo.slidetitle(subj_code, tdir_name)
outname = f"{tdir_name}_{title}.pdf"
```

```
fp, tp = parse_page_range(args.page)
if fp > 0:  # ページ抽出フラグ
    stem, suf = os.path.splitext(outname)
    outname = f"{stem}_test{suf}"
else:
    stem, suf = os.path.splitext(outname)
    if args.tech:
        outname = f"{stem}_tech{suf}"
    elif not args.ho:
        outname = f"{stem}_pr{suf}"

final_pdf = app_dir / outname

# ページ範囲を反映（任意：pypdfでトリミング）
if fp > 0:
    try:
        from pypdf import PdfReader, PdfWriter
        r = PdfReader(str(pdf_path))
        w = PdfWriter()
        start = max(1, fp) - 1
        end = min(len(r.pages), tp)
        for i in range(start, end):
            w.add_page(r.pages[i])
        with open(final_pdf, "wb") as f:
            w.write(f)
    except Exception as e:
        print("ページ抽出に失敗しました。フルPDFをコピーします:",
e)
        shutil.copy2(pdf_path, final_pdf)
else:
    shutil.copy2(pdf_path, final_pdf)

print("PDF名:", final_pdf.name)

# 後片付け（build は残すならこのまま、消すなら shutil.rmtree）
# for ext in
["aux","log","nav","out","snm","toc","vrb","fls","fdb_latexm
k"]:
#     for p in build_dir.glob(f"*.{ext}"):
```

```
#         p.unlink(missing_ok=True)

# オプション：テンプレと content を保存しておくなら build からコピー
# (root /
"main.tex").write_text(main_tex.read_text(encoding="utf-8"),
encoding="utf-8")

slideinfo.slideinfoupdate(subj_code, tdir_name)
```

ポイント

- 両テーマとも `latexmk -lualatex -shell-escape` に統一。
- `-outdir=build` に出力して中間物と成果物を分離。
- タイムアウト・ログ要約・堅牢なエラーハンドリング。
- LaTeX 向けに `@@sdir@@` のパスを `/` 正規化。
- `\end{document}` の重複防止。
- `--page` を **実際に PDF トリミングへ反映**（`pypdf` 使用。不要なら削除可）。

2) テンプレ（SimpleDarkBlue 側）の整理要点

- `xcolor` の二重読み込みをやめる（`[table,xcdraw]{xcolor}` の一箇所に）。
- `hyperref` は**最後**に。
- `footline` 設定は一つに統一（同ファイル内で 2 回書かない）。
- minted 使用前提なら、**ドキュメント内で verbatim を含むフレームは `[fragile]` にするルール**で運用。

（例）

```
\documentclass[aspectratio=169,xcolor=dvipsnames]{beamer}
\usetheme{SimpleDarkBlue}

\usepackage{luatexja}
\usepackage{colortbl}
\usepackage[most]{tcolorbox}
```

```
\usepackage[table,xcdraw]{xcolor} % ← ここだけ
\usepackage{array}
\usepackage{tabularx}
\usepackage{dirtree}
\usepackage{minted}                    % ← minted 使用
\usepackage{tikz}
\usepackage{graphicx}
\graphicspath{{images/}{../@@sdir@@/images/}}
\usepackage{booktabs}
\usepackage{url}
\usepackage{makecell}
\usepackage{lipsum}
\usepackage{setspace}
\usepackage{mathtools}
\usepackage{longtable}
\usepackage{pdfpages}

%  （最後に）
\usepackage[colorlinks=true,linkcolor=blue,urlcolor=blue]
{hyperref}

% footline の定義は一回にまとめる（重複させない）
\newcommand{\myfootertext}{@@sdir@@}
\setbeamertemplate{footline}{%
  \leavevmode
  \hbox to \paperwidth{%
    \hspace*{0.2cm}
    \scriptsize\color{gray!50} \myfootertext
    \hfill
    \scriptsize\color{gray} \insertframenumber{} /
\inserttotalframenumber
    \hspace*{0.4cm}
  }%
  \vspace{1pt}
}

% teacher/pause マクロ（外部 .sty でも可）
\usepackage{teacherframe}
```

```
%@@pausemode@@
%@@teachermode@@

\title{ @@stitle@@ }
\date{}

\newcommand{\codedir}{../@@sdir@@/}

\begin{document}
\begin{frame}[plain,noframenumbering]
  \titlepage
  \bigskip
  \begin{center}
    \ifteachermode 教師用 \fi
  \end{center}
\end{frame}

\setcounter{framenumber}{0}
```

補足：よくある実運用でのハマり

- **Windows での minted**：`-shell-escape` と Python/Pygments が必要。ないと沈黙失敗するので、起動前に存在チェック→警告が親切。
- **日本語や結合文字を含むパス**：LuaLaTeX でも環境差でコケることがあるため、**ビルド系パスは英数字のみ**が無難。
- **長い verbatim コード**：`[fragile]` を忘れると謎エラー。テンプレから `\begin{frame}[fragile]` を出力するユーティリティを用意すると安全。
- **ページ部分出力**：PPT 印刷など用に、`--page` のショートカットが重宝します（今回 pypdf の例を追加済み）。

必要なら、この最適化を前提に **metropolis テンプレの方も同様に整えた案**をお出しします。