

# Manga

Bash script to download mangas from mangaread.org

[mangaread-downloader.sh](#)

```
#!/usr/bin/env bash
#
# mangareader-downloader.sh
# Downloader for mangaread.org
#
# Usage: ./mangareader-downloader.sh [-v] [-f] [-j N] [-r RETRIES]
# <manga-url-or-name-or-url> [chapter-range]
# Options:
# -v          verbose
# -f          force (re-download chapters; deletes chapter folder
# before download)
# -j N        concurrency (parallel jobs, default 3)
# -r N        retries per image (default 3)
#
set -euo pipefail

VERBOSE=0
FORCE=0
JOBS=3
RETRIES=3
UA="Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/121.0 Safari/537.36"

usage() {
    cat <<USAGE
Usage: $0 [-v] [-f] [-j N] [-r RETRIES] <manga-url-or-name-or-url>
[chapter-range]
Examples:
    $0 the-beginning-after-the-end
    $0 -v -j 4 -r 3 the-beginning-after-the-end 84
    $0 -f the-beginning-after-the-end 84
USAGE
    exit 1
}

while getopts ":vfj:r:" opt; do
    case ${opt} in
        v ) VERBOSE=1 ;;
        f ) FORCE=1 ;;
        j ) JOBS="$OPTARG" ;;
        r ) RETRIES="$OPTARG" ;;
        \? ) echo "Invalid option: -$OPTARG" >&2; usage ;;
    esac
done
```

```

done
shift $((OPTIND-1))

if [[ $# -lt 1 ]]; then usage; fi
INPUT="$1"
CHAP_RANGE="${2:-}"

logv() { [[ $VERBOSE -eq 1 ]] && echo "$@"; }
err() { echo "ERROR: $@" >&2; }

slugify() {
    echo "$1" | tr '[:upper:]' '[:lower:]' | sed -E 's/[^a-z0-9]+/-/g;
s/^+|-+$/g'
}

norm_url() {
    local u="$1"
    if [[ "$u" =~ ^/ ]]; then
        echo "https://www.mangaread.org${u}"
    else
        echo "$u"
    fi
}

# Determine manga URL & slug
if [[ "$INPUT" =~ ^https?:// ]]; then
    MANGA_URL="$INPUT"
    [[ "${MANGA_URL: -1}" != "/" ]] && MANGA_URL="${MANGA_URL}/"
    SLUG=$(echo "$MANGA_URL" | sed -E 's:/*$::' | awk -F/ '{print $NF}')
else
    SLUG=$(slugify "$INPUT")
    MANGA_URL="https://www.mangaread.org/manga/$SLUG/"
fi

echo "Fetching manga page: $MANGA_URL"
HTML=$(curl -sL --fail -A "$UA" "$MANGA_URL" 2>/dev/null) || {
    err "Failed to fetch $MANGA_URL - check URL or network."
    exit 1
}

# find <li> blocks referencing /manga/<slug>/chapter
LI_BLOCKS=$(echo "$HTML" | tr '\n' ' ' | sed -E
's/<li([\^>]*)>/\n<li\1>/g' | grep -i "/manga/$SLUG/chapter" || true)
if [[ -z "$LI_BLOCKS" ]]; then
    err "No chapter blocks found for '/manga/$SLUG/chapter'. The page
layout may have changed."
    exit 1
fi

declare -A CHAP_URL CHAP_TITLE CHAP_DATE
CHAP_ORDER=()

```

```

while IFS= read -r li; do
  href=$(echo "$li" | grep -oP 'href\s*=\s*["'\`']?[^"'\`']>]*"$SLUG"/chapter[^"'\`']>*"["'\`']?' | head -n1 || true)
  href=${href#href=} ; href=${href#\} ; href=${href#\} ;
href=${href%\} ; href=${href%\} 2>/dev/null || true
  [[ -z "$href" ]] && continue
  url=$(norm_url "$href")

  title=$(echo "$li" | grep -oP '<a[^>]*.*?</a>' | sed -E
's/<[^>]*>\/g' | sed -E 's/^[[:space:]]+|[[:space:]]+$/g' | head -n1)
  date=$(echo "$li" | grep -oP '\d{1,2}\.\d{1,2}\.\d{4}' | head -n1 ||
true)

  if [[ "$title" =~ [Cc]hapter[[:space:]]*([0-9]+(\.[0-9]+)?) ]]; then
    chap="${BASH_REMATCH[1]}"
  else
    chap="unknown-$(printf '%03d' $((RANDOM%900+100)))"
  fi

  if [[ -z "${CHAP_URL[$chap]:-}" ]]; then
    CHAP_URL[$chap]=$url
    CHAP_TITLE[$chap]=$title
    CHAP_DATE[$chap]=$date
    CHAP_ORDER+=("$chap")
    logv "Found chapter: $chap -> $url (${date:-no-date})"
  fi
done <<< "$LI_BLOCKS"

if [[ ${#CHAP_ORDER[@]} -eq 0 ]]; then
  err "No chapters parsed from page."
  exit 1
fi

ALL_CHAPS=$(printf "%s\n" "${CHAP_ORDER[@]}" | sort -V -u)

# Determine SELECTED chapters based on CHAP_RANGE (supports ranges like
10-20, decimals, comma list)
SELECTED=()
if [[ -n "$CHAP_RANGE" ]]; then
  if [[ "$CHAP_RANGE" =~ ^[0-9]+(\.[0-9]+)?-[0-9]+(\.[0-9]+)?$ ]]; then
    START=${CHAP_RANGE%-*}
    END=${CHAP_RANGE#*-}
    in_range() { awk -v v="$1" -v a="$2" -v b="$3" 'BEGIN{ if((v+0)>=
(a+0) && (v+0) <= (b+0)) exit 0; else exit 1 }'; }
    for c in "${ALL_CHAPS[@]"; do
      if [[ "$c" =~ ^unknown- ]]; then continue; fi
      if in_range "$c" "$START" "$END"; then SELECTED+=("$c"); fi
    done
  elif [[ "$CHAP_RANGE" == *,,* ]]; then
    IFS=',' read -ra items <<< "$CHAP_RANGE"

```

```

    for it in "${items[@]}"; do
        it=$(echo "$it" | sed 's/^[[:space:]]*///;s/[[:space:]]*$//')
        [[ -n "${CHAP_URL[$it]:-}" ]] && SELECTED+=("$it") || logv
    "Requested chapter $it not found (skipping)"
    done
    else
        if [[ -n "${CHAP_URL[$CHAP_RANGE]:-}" ]]; then
    SELECTED+=("$CHAP_RANGE"); else err "Requested chapter '$CHAP_RANGE'
    not found."; exit 1; fi
        fi
    else
        SELECTED=("${ALL_CHAPS[@]}")
    fi

if [[ ${#SELECTED[@]} -eq 0 ]]; then err "No chapters selected (after
    applying range/filter)."; exit 1; fi

OUTDIR="$SLUG"
mkdir -p "$OUTDIR"

# check for GNU parallel
if command -v parallel >/dev/null 2>&1; then
    PARALLEL_AVAILABLE=1
else
    PARALLEL_AVAILABLE=0
    logv "GNU parallel not found – falling back to sequential downloads."
fi

# download_chapter: fetch a single chapter page (no pagination) and
# download all <img class="wp-manga-chapter-img">
download_chapter() {
    local chap="$1"
    local url="${CHAP_URL[$chap]}"

    # create padded folder name: integer -> zero-pad to 3 digits;
    # decimals keep fraction
    local safechap
    if [[ "$chap" =~ ^([0-9]+)(\.[0-9]+)?$ ]]; then
        ip="${BASH_REMATCH[1]}; fp="${BASH_REMATCH[2]}"
        safechap=$(printf "%03d%s" "$ip" "$fp")
    else
        safechap="$chap"
    fi
    local folder="$OUTDIR/chapter-$safechap"

    echo "=== Chapter $chap: ${CHAP_TITLE[$chap]:-N/A}
    (${CHAP_DATE[$chap]:-no-date}) ==="
    logv "Chapter URL: $url"

    if [[ $FORCE -eq 1 ]]; then rm -rf "$folder"; fi
    mkdir -p "$folder"

```

```

if [[ -n "$(ls -A "$folder" 2>/dev/null)" && $FORCE -eq 0 ]]; then
    echo "⏏ Skipping chapter $chap – folder '$folder' not empty
(resume)."
    return
fi

pagehtml=$(curl -sL --fail -A "$UA" "$url" 2>/dev/null) || {
    err "Failed to fetch chapter page: $url"
    return
}

# Normalize whitespace to catch src attributes split across lines
cleanhtml=$(echo "$pagehtml" | tr '\n\r\t' ' ' | sed -E 's/[ ]+/ /g')

# Extract image URLs only from wp-manga-chapter-img
mapfile -t imgs <<(
    echo "$cleanhtml" |
        grep -oP '<img[>]+class="[^"]*wp-manga-chapter-img[^"]*" [^>]*>'
|
        sed -E 's/.*(data-src|data-lazy-src|data-
original|src)=[''\''']([^\''\''']+)[''\'''].*\/\2/'
)

if [[ ${#imgs[@]} -eq 0 ]]; then
    err "No chapter images found for chapter $chap (parsing failed or
layout changed)."
    return
fi

echo "Found ${#imgs[@]} images. Downloading (concurrency: $JOBS,
retries: $RETRIES)..."

TMPFILE=$(mktemp)
idx=1
for u in "${imgs[@]}; do
    # make sure to trim any stray whitespace (this is the critical fix)
    u=$(printf "%s" "$u" | sed -E 's/^[[:space:]]+//;
s/[[:space:]]+$//; s/~/\&/g')
    [[ -z "$u" ]] && continue

    ext=$(echo "${u##*}" | sed -E 's/\?.*$//' | tr '[:upper:]'
'[:lower:]')
    if ! [[ "$ext" =~ ^(jpg|jpeg|png|webp)$ ]]; then ext="jpg"; fi
    fname=$(printf "%03d.%s" "$idx" "$ext")
    out="$folder/$fname"
    printf "%d\t%s\t%s\n" "$idx" "$u" "$out" >> "$TMPFILE"
    idx=$((idx+1))
done

if [[ $PARALLEL_AVAILABLE -eq 1 ]]; then
    parallel -j "$JOBS" --colsep $'\t' \

```

```

' bash -c '\''idx="$1"; url="$2"; out="$3";
  if [[ -f "$out" ]]; then
    echo "  □ $out already exists";
  else
    attempts=0; ok=0;
    while [[ $attempts -lt "$RETRIES" ]]; do
      attempts=$((attempts+1));
      echo "  ↓□ $out (attempt $attempts)";
      if curl -sL --fail -A "$UA" "$url" -o "$out"; then ok=1;
break; else echo "  □ attempt $attempts failed for $url" >&2; sleep 1;
fi;
      done;
      if [[ $ok -ne 1 ]]; then echo "  □ All attempts failed for
$url" >&2; rm -f "$out"; fi;
    fi
    '\'' _ {1} {2} {3}' :::: "$TMPFILE"
else
# sequential fallback
while IFS=$'\t' read -r idx url out; do
  if [[ -f "$out" ]]; then
    echo "  □ $out already exists"
    continue
  fi
  attempts=0; ok=0
  while [[ $attempts -lt $RETRIES ]]; do
    attempts=$((attempts+1))
    echo "  ↓□ $out (attempt $attempts)"
    if curl -sL --fail -A "$UA" "$url" -o "$out"; then ok=1; break;
else echo "  □ attempt $attempts failed for $url" >&2; sleep 1; fi
  done
  if [[ $ok -ne 1 ]]; then echo "  □ All attempts failed for $url"
>&2; rm -f "$out"; fi
  done < "$TMPFILE"
fi

rm -f "$TMPFILE"
echo "  □ Finished chapter $chap"
}

TOTAL=${#SELECTED[@]}
i=0
for chap in "${SELECTED[@]"; do
  i=$((i+1))
  echo ""
  echo ">> ($i/$TOTAL) processing chapter $chap"
  download_chapter "$chap"
  sleep $((1 + RANDOM % 2))
done

echo ""

```

```
echo "All done – saved under: $OUTDIR"
```

From:

<http://wuff.dyndns.org/> - **Wulf's Various Things**

Permanent link:

<http://wuff.dyndns.org/doku.php?id=howto:manga&rev=1758882146>

Last update: **2025/09/26 11:22**

