1 Goal

We build a BLAST nucleotide database for an arbitrary target group (e.g., `Ustilaginaceae`). Steps:

1. Resolve the target group to its accessions using `taxi` and `neighbors`.
2. Download the corresponding genomes via `ncbi-datasets` CLI.
3. Map sequence IDs to NCBI TaxIDs using the `neidb` SQLite DB.
4. Concatenate FASTAs and create a `makeblastdb` with `-taxid_map`.

Dependencies: `taxi`, `neighbors`, `sqlite3`, `datasets` (NCBI CLI), `unzip`, `awk`, `grep`, `tr`, `makeblastdb`.

2 Top-level script

This chunk defines inputs, derived filenames, a temporary workspace, and basic sanity checks, then executes the pipeline by invoking the step chunks in order:

1. **Inputs & names:** Read the target group and the path to `neidb`, derive a sanitized prefix for all outputs.
2. **Workspace:** Establish paths for accessions, mapping, final FASTA, and BLAST DB; create a temporary unzip directory with cleanup.
3. **Checks:** Verify required executables and the presence of `neidb`.
4. **Execution flow:** Run ⟨get-accessions⟩, then ⟨download-genomes⟩, ⟨build-mapping⟩, ⟨build-taxid-map⟩, ⟨concatenate-fasta⟩, and finally ⟨make-blastdb⟩.

⟨*main*⟩≡
```
#!/usr/bin/env bash
set -euo pipefail

TARGET_GROUP="${1:-}"
NEIDB_PATH="${2:-${HOME}/DBs/neidb}"

if [[ -z "${TARGET_GROUP}" ]]; then
  echo "Usage: $0 \"<target group>\" [path/to/neidb]"
  exit 1
fi

sanitize() {
  tr '[:upper:]' '[:lower:]' | tr -cs '[:alnum:]' '_' | sed 's/^_*//; s/_*$//'
}
PREFIX="$(printf '%s' "${TARGET_GROUP}" | sanitize)"
```

```
WORKDIR="$(pwd)"
ACC_TXT="${WORKDIR}/${PREFIX}.accessions.txt"
ACC_TSV="${WORKDIR}/${PREFIX}.accsIDs.tsv"
TAXID_MAP="${WORKDIR}/${PREFIX}.taxid_map.txt"
DB_FASTA="${WORKDIR}/${PREFIX}.fna"
DB_TITLE="${PREFIX}Db"
DB_PREFIX="${PREFIX}Db"

TMPDIR="$(mktemp -d "${WORKDIR}/tmp_${PREFIX}_XXXXXX")"
cleanup() { rm -rf "${TMPDIR}"; }
trap cleanup EXIT

need() { command -v "$1" >/dev/null 2>&1 || { echo "Missing dependency: $1" >&2; exit 1
for bin in taxi neighbors sqlite3 datasets unzip awk grep tr makeblastdb; do need "$bin
[[ -f "${NEIDB_PATH}" ]] || { echo "neidb not found at: ${NEIDB_PATH}" >&2; exit 1; }
```

⟨*get-accessions*⟩

⟨*download-genomes*⟩

⟨*build-mapping*⟩

⟨*build-taxid-map*⟩

⟨*concatenate-fasta*⟩

⟨*make-blastdb*⟩

3 Get accessions via `taxi` + `neighbors`

This step resolves the user's target group to a list of genome accessions. We ask `taxi` for the group identifier, pass it to `neighbors -g` to obtain related genome rows, keep rows tagged `tt` (targets), extract the final, pipe-separated accession field, split it into one accession per line, and write the result to a text file for downstream steps.

⟨*get-accessions*⟩≡
```
taxi "${TARGET_GROUP}" "${NEIDB_PATH}" \
  | awk 'NR==2{print $1}' \
  | neighbors -g "${NEIDB_PATH}" \
  | awk '$1=="tt" {print $NF}' \
  | tr '|' '\n' \
  | sed '/^$/d' \
  > "${ACC_TXT}"

echo "  → wrote $(wc -l < "${ACC_TXT}") accessions to ${ACC_TXT}"
```

4 Download genomes with `datasets`

Here we iterate over each accession, fetch the NCBI dataset archive, unzip it into a temporary directory, move any `.fna` files up to the working directory, and remove the temporary files. If a bundle lacks `.fna`, we quietly skip it.

⟨*download-genomes*⟩≡
```
while IFS= read -r accession; do
  zipf="${TMPDIR}/${accession}.zip"
  datasets download genome accession "${accession}" --filename "${zipf}" >/dev/null 2>&
  unzip -o "${zipf}" -d "${TMPDIR}/unz_${accession}" >/dev/null
  fna_dir="${TMPDIR}/unz_${accession}/ncbi_dataset/data/${accession}"
  if compgen -G "${fna_dir}/*.fna" > /dev/null; then
    mv "${fna_dir}/"*.fna "${WORKDIR}/"
  fi
  rm -rf "${TMPDIR}/unz_${accession}" "${zipf}"
done < "${ACC_TXT}"

echo "  → current *.fna count: $(ls -1 *.fna 2>/dev/null | wc -l || true)"
```

5 Accession → TaxID table from `neidb`

For each accession, we query `neidb`'s `genome` table to retrieve `taxid|accession` pairs and normalize to TSV. This provides the taxonomic identifier associated with each accession and is the basis for building the `makeblastdb -taxid_map`.

⟨*build-mapping*⟩≡
```
: > "${ACC_TSV}"
while IFS= read -r a; do
  sqlite3 "${NEIDB_PATH}" \
    "select taxid,accession from genome where accession like '${a}'" \
    | tr '|' '\t' >> "${ACC_TSV}"
done < "${ACC_TXT}"

sed -i '' -e '/^[[:space:]]*$/d' "${ACC_TSV}" 2>/dev/null || sed -i '/^[[:space:]]*$/d

echo "  → wrote $(wc -l < "${ACC_TSV}") rows to ${ACC_TSV}"
```

6 SequenceID → TaxID map for `makeblastdb`

`makeblastdb -taxid_map` expects lines of the form `<sequence_id><TAB><taxid>`. We scan all FASTA headers for files matching each accession, take the first header token (without the leading `>`) as the sequence ID, pair it with the corresponding TaxID from the TSV, and de-duplicate the result.

⟨*build-taxid-map*⟩≡
```
: > "${TAXID_MAP}"
while IFS=$'\t' read -r tax acc; do
  for f in *"${acc}"*.fna; do
    [[ -e "$f" ]] || continue
    grep -h "^>" "$f" \
      | awk -v t="${tax}" '{
            id=$1; sub(/^>/,"",id);
            print id "\t" t
         }' \
      >> "${TAXID_MAP}"
  done
done < "${ACC_TSV}"

sort -u -o "${TAXID_MAP}" "${TAXID_MAP}"
echo "  → taxid_map lines: $(wc -l < "${TAXID_MAP}")"
```

7 Concatenate FASTAs

We concatenate all `.fna` files into a single FASTA for database creation and fail fast if nothing was produced.

⟨*concatenate-fasta*⟩≡
```
rm -f "${DB_FASTA}"
shopt -s nullglob
for fna in *.fna; do
  cat "${fna}" >> "${DB_FASTA}"
  rm -f "${fna}"
done
shopt -u nullglob

if [[ ! -s "${DB_FASTA}" ]]; then
  echo "ERROR: ${DB_FASTA} is empty; no FASTAs were found/concatenated." >&2
  exit 1
fi
```

8 Build the BLAST DB

We create a nucleotide database with parsed sequence IDs and the explicit `taxid_map`, giving the database a descriptive title and prefix.

⟨*make-blastdb*⟩≡
```
makeblastdb \
  -in "${DB_FASTA}" \
  -dbtype nucl \
  -parse_seqids \
  -taxid_map "${TAXID_MAP}" \
  -title "${DB_TITLE}" \
  -out "${DB_PREFIX}"
```