```
#!/bin/bash
# build_repo.sh - Build a custom pacman repository from your package
lists.

# --- Color Definitions ---
export NC='\033[0m'
export WHITE='\033[1;37m'
export RED='\033[0;31m'
export PASTEL_TEAL='\033[1;36m'
export PASTEL_PURPLE='\033[1;35m'
export PASTEL_LIGHTBLUE='\033[1;34m'
export PASTEL_LIGHTGREEN='\033[1;32m'
export PASTEL_YELLOW='\033[1;33m'

# --- Configuration ---
REPO_DIR="./my_custom_repo"
REPO_NAME="my_custom_repo"
AUR_LIST="data/aur.list"
PKG_LIST="data/packages.list"
BUILD_DIR="/tmp/repo_builder"

# --- Infrastructure ---

show_header() {
    clear
    echo -e "${PASTEL_TEAL}"
    echo " .d88b.  d8888b. d8888b.      d8888b. db    db d88888b
d88888b"
    echo " .8P  Y8. 88  \`8D 88  \`8D      88  \`8D 88    88 88'
88'    "
    echo " 88    88 88   88 88   88      88oooY' 88    88 88fffff
88fffff"
    echo " 88    88 88   88 88   88      88~~~b. 88    88 88.     88.
"
    echo " \`8b  d8' 88  .8D 88  .8D      88   8D 88b  d88 88.     88.
"
    echo "  \`Y88P'  Y8888D' Y8888D'      Y8888P' ~Y8888P' Y88888P
Y88888P"
    echo -e "${NC}"
    echo -e
"${PASTEL_LIGHTBLUE}================================================
================"
    echo -e "${WHITE}        Custom Arch Repository Builder & Offline
Sync Tool        ${NC}"
    echo -e
"${PASTEL_LIGHTBLUE}================================================
================"
    echo
```

```
}

check_dependencies() {
    local deps=("git" "base-devel" "pacman-contrib")
    local missing=()
    for dep in "${deps[@]}"; do
        if ! pacman -Qi "$dep" &> /dev/null; then
            missing+=("$dep")
        fi
    done

    if [ ${#missing[@]} -gt 0 ]; then
        echo -e "${PASTEL_YELLOW}Installing missing dependencies for
building: ${missing[*]}${NC}"
        sudo pacman -S --noconfirm --needed "${missing[@]}"
    fi

    # Check for yay (needed for fetching AUR PKGBUILDS easily)
    if ! command -v yay &> /dev/null; then
        echo -e "${RED}Error: 'yay' is required to fetch AUR
sources.${NC}"
        echo -e "${WHITE}Please run ./modules/02-aur.sh or install yay
manually first.${NC}"
        exit 1
    fi
}

# --- Build Functions ---

setup_environment() {
    echo -e "${PASTEL_TEAL}Setting up workspace...${NC}"
    mkdir -p "$REPO_DIR"
    mkdir -p "$BUILD_DIR"

    # Clean build dir
    rm -rf "$BUILD_DIR/*"
}

build_aur_packages() {
    echo -e "\n${PASTEL_PURPLE}--- Phase 1: Building AUR Packages
---${NC}"

    if [ ! -f "$AUR_LIST" ]; then
        echo -e "${RED}AUR list not found at $AUR_LIST${NC}"
        return
    fi

    local current_dir=$(pwd)
```

```
    while read -r pkg; do
        [[ -z "$pkg" ]] && continue

        # Check if already in repo
        if ls "$REPO_DIR/$pkg"*.pkg.tar.zst 1> /dev/null 2>&1; then
            echo -e "${PASTEL_LIGHTGREEN}Skipping $pkg (Already exists
in repo)${NC}"
            continue
        fi

        echo -e "${PASTEL_LIGHTBLUE}Fetching and Building:
${WHITE}$pkg${NC}"

        cd "$BUILD_DIR" || exit

        # 1. Fetch PKGBUILD
        if yay -G "$pkg" > /dev/null 2>&1; then
            cd "$pkg" || continue

            # 2. Build (Make package, sync deps, install needed deps
for build, noconfirm)
            # We assume user has sudo rights.
            if makepkg -s --noconfirm --clean; then
                echo -e "${PASTEL_LIGHTGREEN}Successfully built
$pkg${NC}"
                mv *.pkg.tar.zst "$current_dir/$REPO_DIR/"
            else
                echo -e "${RED}Failed to build $pkg${NC}"
            fi
        else
            echo -e "${RED}Could not fetch $pkg from AUR.${NC}"
        fi

        cd "$current_dir" || exit

    done < "$AUR_LIST"
}

download_official_packages() {
    echo -e "\n${PASTEL_PURPLE}--- Phase 2: Downloading Official
Packages ---${NC}"

    if [ ! -f "$PKG_LIST" ]; then
        echo -e "${RED}Package list not found at $PKG_LIST${NC}"
        return
    fi
```

```
    # Read list into array
    mapfile -t pkgs < "$PKG_LIST"

    # Filter empty lines
    local valid_pkgs=()
    for p in "${pkgs[@]}"; do
        [[ -n "$p" ]] && valid_pkgs+=("$p")
    done

    echo -e "${WHITE}Downloading ${#valid_pkgs[@]} packages to
repository cache...${NC}"

    # Use pacman to download (-w) to our specific directory
(--cachedir)
    # This grabs the package AND its dependencies if they aren't
installed?
    # No, -Sdw downloads target + deps. -Sw just target.
    # We usually want just the targets listed, or targets+deps for
full offline repo.
    # We will use simple download for listed packages.

    sudo pacman -Sw --cachedir "$REPO_DIR" --noconfirm --needed
"${valid_pkgs[@]}"

    # Note: official packages are signed. repo-add works fine with
them.
}

create_repository_db() {
    echo -e "\n${PASTEL_PURPLE}--- Phase 3: Generating Repository
Database ---${NC}"

    cd "$REPO_DIR" || exit

    # Find all packages in the dir
    local pkg_files=(*.pkg.tar.zst)

    if [ ${#pkg_files[@]} -eq 0 ]; then
        echo -e "${PASTEL_YELLOW}No packages found in repo
directory.${NC}"
        return
    fi

    echo -e "${PASTEL_TEAL}Adding ${#pkg_files[@]} packages to
database '$REPO_NAME.db.tar.gz'...${NC}"

    # repo-add updates the DB with the packages
    # -n: New packages only (faster)
```

```
    # -R: Remove old entries if file is gone
    repo-add -n -R "$REPO_NAME.db.tar.gz" *.pkg.tar.zst

    cd ..
}

show_instructions() {
    local abs_path=$(realpath "$REPO_DIR")

    echo
    echo -e "${PASTEL_PURPLE}
╭──────────────────────────────────────────────────────────────────────────╮${NC
}"
    echo -e "${PASTEL_PURPLE} |
|${NC}"
    echo -e "${PASTEL_PURPLE} |   ${PASTEL_LIGHTGREEN}
Repository Build Complete!                    ${PASTEL_PURPLE}
|${NC}"
    echo -e "${PASTEL_PURPLE} |
|${NC}"
    echo -e "${PASTEL_PURPLE}
╰──────────────────────────────────────────────────────────────────────────╯${NC
}"
    echo
    echo -e "${WHITE}Your repository is located at:${NC}
${PASTEL_TEAL}$abs_path${NC}"
    echo
    echo -e "${WHITE}To use this repository, add the following to your
${PASTEL_YELLOW}/etc/pacman.conf${WHITE}:${NC}"
    echo
    echo -e "${PASTEL_LIGHTBLUE}[$REPO_NAME]${NC}"
    echo -e "${PASTEL_LIGHTBLUE}SigLevel = Optional TrustAll${NC}"
    echo -e "${PASTEL_LIGHTBLUE}Server = file://$abs_path${NC}"
    echo
    echo -e "${WHITE}(Note: 'SigLevel = Optional TrustAll' is needed
because custom builds aren't signed by Arch)${NC}"
}

# --- Main ---
show_header

# Safety check for root (don't run makepkg as root)
if [ "$EUID" -eq 0 ]; then
   echo -e "${RED}Please do not run this script as root.${NC}"
   echo -e "${WHITE}Building packages requires a normal user. The
script will ask for sudo when needed.${NC}"
   exit 1
fi
```

```
check_dependencies
setup_environment
build_aur_packages
download_official_packages
create_repository_db
show_instructions
```