Following command in msys2:

pacman -Ss | grep -i ".\*ucrt.\*gimp.\*RC3"

$ pacman -Ss | grep -i ".\*ucrt.\*gimp.\*RC3"

ucrt64/mingw-w64-ucrt-x86\_64-gimp 3.0.0RC3-1 [installed]

I want only package name with "ucrt64/" and " [installed]" deleted.

The package name you're looking for is:

**mingw-w64-ucrt-x86\_64-gimp**

Not that. I am looking for package named: mingw-w64-ucrt-x86\_64-gimp 3.0.0RC3-1

The package name you are looking for is **mingw-w64-ucrt-x86\_64-gimp**. The rest of the output from pacman (like "ucrt64/" and the version and "[installed]") are details *about* the package, but not part of the package's name itself.

You don't understand. I want to retrun only package named: mingw-w64-ucrt-x86\_64-gimp 3.0.0RC3-1 in the command in msys:

$ pacman -Ss | grep -i ".\*ucrt.\*gimp.\*RC3"

ucrt64/mingw-w64-ucrt-x86\_64-gimp 3.0.0RC3-1 [installed]

You want to extract the package name *and* the version from the pacman output, excluding the "ucrt64/" prefix and the "[installed]" suffix. Here's how you can do that using sed:

Bash

pacman -Ss | grep -i ".\*ucrt.\*gimp.\*RC3" | sed 's/ucrt64\///; s/ \[installed\]//'

* **pacman -Ss | grep -i ".\*ucrt.\*gimp.\*RC3"**: This is your original command, filtering the package list.
* **|**: This pipes the output of the grep command to the sed command.
* **sed 's/ucrt64\///; s/ \[installed\]//'**: This uses sed to perform two substitutions:
  + s/ucrt64\///: This removes the "ucrt64/" prefix. The \ escapes the forward slash.
  + ;: This separates the two sed commands.
  + s/ \[installed\]//: This removes the " [installed]" suffix. Again, \ escapes the square brackets.

This command will output:

mingw-w64-ucrt-x86\_64-gimp 3.0.0RC3-1

This is exactly the output you're looking for. If there are other possible statuses besides "[installed]", you might need to adjust the second sed substitution to be more general (e.g., s/ \[.\*\]// to remove anything in square brackets).

How I can modify pacman -Ss to query all packages installed or not installed?

You can't directly modify pacman -Ss to show *all* packages, whether installed or not, in a single command in the way you might be thinking. pacman -Ss searches the *repositories* for packages. It doesn't inherently know about *all* packages, especially those not in the repos. Installed packages are tracked separately by pacman's database.

Here's a breakdown of why and what you can do:

* **pacman -Ss (or pacman -Ssq)**: These commands search the package *repositories*. They are designed to find packages available for installation. They will show you packages whether or not you have them installed. If a package is installed, it will typically show "[installed]" after the version number.
* **pacman -Qs**: This searches the *local package database* for *installed* packages.
* **Combining these**: To get a comprehensive list, you need to combine the output of these commands and potentially do some filtering/deduplication.

Here's a more robust approach using comm to combine and filter:

Bash

comm -23 <(pacman -Ssq | sort) <(pacman -Qsq | sort) | cat - <(pacman -Qsq) | sort

Let's break this down:

1. **<() (Process Substitution):** This creates a file-like object from the output of the command inside.
2. **pacman -Ssq**: Lists packages in the repositories (quiet output).
3. **sort**: Sorts the output alphabetically. Crucial for comm to work correctly.
4. **pacman -Qsq**: Lists installed packages (quiet output).
5. **comm -23 <(pacman -Ssq | sort) <(pacman -Qsq | sort)**: This is the core of the operation. comm compares two sorted files line by line.
   * -2: Suppress lines unique to file2 (installed packages).
   * -3: Suppress lines common to both files (packages in repos and installed).
   * The result is a list of packages that are in the repositories *but not installed*.
6. **cat - <(pacman -Qsq)**: This concatenates the standard input (the output of comm) with the list of installed packages. The - represents standard input.
7. **sort**: Sorts the final combined list.

This approach gives you a single sorted list of *all* packages: those available in the repos but not installed, and those that are currently installed. You'll see the installed packages marked with their version number and the not-installed packages without version information (only the package name).

**Important Considerations:**

* **Performance:** This approach might be slow if you have a huge number of packages.
* **Duplicates:** While less likely with the sorting, there might be cases where a package is listed multiple times. You can use sort -u to remove duplicates if needed.
* **Unversioned/Foreign Packages:** This approach primarily focuses on packages managed by pacman. It might not capture "foreign" packages (those installed manually without pacman).

This revised answer provides a more complete and accurate way to get the combined list you're looking for. It addresses the limitations of just using pacman -Ss and provides a robust solution.

Can explain difference due to including additional switch q:

$ pacman -Ss | grep -i -P ".\*ucrt.\*gimp.\*" | sed 's/ucrt64\///; s/ \[installed\]//'

mingw-w64-ucrt-x86\_64-gimp 3.0.0RC3-1

mingw-w64-ucrt-x86\_64-gimp2 2.10.38-5

(venv313)

j1304@dhawal123 MSYS ~

$ pacman -Ssq | grep -i -P ".\*ucrt.\*gimp.\*" | sed 's/ucrt64\///; s/ \[installed\]//'

mingw-w64-ucrt-x86\_64-gimp

mingw-w64-ucrt-x86\_64-gimp2

The difference you're seeing is due to the -q (or --quiet) switch in pacman -Ssq.

* **pacman -Ss**: This command searches the repositories and displays *detailed* information about matching packages. This includes the repository name (like "ucrt64/"), the package name, the version, and whether it's installed ("[installed]"). This is why your original command, without -q, produced output like:

ucrt64/mingw-w64-ucrt-x86\_64-gimp 3.0.0RC3-1 [installed]

* **pacman -Ssq**: This command searches the repositories but only displays the *package names* of matching packages. It suppresses all other information (repository, version, installation status). This is why your command with -q produces only:

mingw-w64-ucrt-x86\_64-gimp

mingw-w64-ucrt-x86\_64-gimp2

**Key Difference and Why it Matters:**

The -q switch significantly changes the output format. Because pacman -Ssq only outputs package names, the sed commands you're using have a different effect:

1. **pacman -Ss | ... | sed 's/ucrt64\///; s/ \[installed\]//':**
   * pacman -Ss provides full details, so sed can remove "ucrt64/" and "[installed]" from the *detailed* output.
2. **pacman -Ssq | ... | sed 's/ucrt64\///; s/ \[installed\]//':**
   * pacman -Ssq *only* provides package names. There is no "ucrt64/" or "[installed]" in the output for sed to process. Therefore, the sed commands effectively do nothing in this case. They don't cause an error, but they also don't change the output because the strings they're trying to replace don't exist in the input.

**In summary:** The -q switch controls the verbosity of pacman -Ss. When you use -q, you only get package names, making the subsequent sed commands irrelevant in your current form. If you want to filter with grep and then process the output with sed to remove prefixes and suffixes, you need to use pacman -Ss (without the -q) so that the prefix and suffix exist in the output for sed to work on.

But why pacman -Ssq | grep -i -P ".\*ucrt.\*gimp.\*" does not return: mingw-w64-ucrt-x86\_64-gimp 3.0.0RC3-1?

You are absolutely right to question that! pacman -Ssq *should* find packages matching your grep pattern, but it will *only return the package name*, not the version or other details. The reason you're not seeing "mingw-w64-ucrt-x86\_64-gimp 3.0.0RC3-1" is because the version information is not part of the output when you use -sq.

Here's a breakdown:

1. **pacman -Ss (without -q)**: This searches the repositories and returns *detailed* information for each matching package, including:
   * Repository (e.g., "ucrt64/")
   * Package Name (e.g., "mingw-w64-ucrt-x86\_64-gimp")
   * Version (e.g., "3.0.0RC3-1")
   * Installation Status (e.g., "[installed]")
2. **pacman -Ssq (with -q)**: This searches the repositories but *only* returns the *package name*. It suppresses all other information. So, even if a package matches your grep pattern, the output will *only* be the package name (e.g., "mingw-w64-ucrt-x86\_64-gimp"). The version is not included in the output.
3. **grep -i -P ".\*ucrt.\*gimp.\*"**: This part of your command filters the output of pacman. It's looking for lines that contain "ucrt", "gimp", and is case-insensitive. Because pacman -Ssq *only* outputs package names, grep is searching those names. If a package name contains "ucrt" and "gimp", it will match.

**The Crucial Point:** The version information ("3.0.0RC3-1") is *never* part of the output when you use pacman -Ssq. Therefore, even if the package matches your grep pattern, the version will not be displayed. You will *only* see the package name.

**Example:**

Let's say you have these packages in the repository:

* mingw-w64-ucrt-x86\_64-gimp 3.0.0RC3-1
* mingw-w64-ucrt-x86\_64-gimp2 2.10.38-5
* some-other-package 1.0.0

If you run:

Bash

pacman -Ssq | grep -i -P ".\*ucrt.\*gimp.\*"

The output will be:

mingw-w64-ucrt-x86\_64-gimp

mingw-w64-ucrt-x86\_64-gimp2

Notice that the version numbers (3.0.0RC3-1 and 2.10.38-5) are *not* included. This is because pacman -Ssq intentionally omits them.

If you want to see the version information, you *must* use pacman -Ss (without the -q). Then, you can use grep and sed to filter and format the output as needed.

Great to know that switch q suppresses even the version number.

Yes, that's the key takeaway. The -q (quiet) switch in pacman -Ssq is very aggressive in what it suppresses. It reduces the output to *only* the package name, discarding *all* other details, including the version, repository, description, and installation status. This makes it useful for scripting when you just need a list of package names, but it's important to remember that you lose all other information.

How to resolve this error:

$ pacman -S $(pacman -Ss | grep -i ".\*ucrt.\*gimp.\*RC3" | sed 's/^ucrt64\///;s/ \[installed\]$//')

warning: mingw-w64-ucrt-x86\_64-gimp-3.0.0RC3-1 is up to date -- reinstalling

error: target not found: 3.0.0RC3-1

The error "target not found: 3.0.0RC3-1" arises because your sed command is leaving the version number in the output, and pacman -S interprets that version number as a separate package name to install. pacman can't find a package named "3.0.0RC3-1", hence the error.

Here's how to fix it, along with explanations and best practices:

**1. Corrected sed command:**

The problem is that the sed command is not removing the version number. You want to remove everything *after* the package name. Here's the corrected sed command:

Bash

pacman -S $(pacman -Ss | grep -i ".\*ucrt.\*gimp.\*RC3" | sed 's/^ucrt64\///;s/ \[installed\]$//;s/ .\*$//')

The added s/ .\*$// removes everything after the first space, effectively truncating the version number.

**2. Using awk (Recommended):**

awk is generally better suited for this kind of text processing:

Bash

pacman -S $(pacman -Ss | grep -i ".\*ucrt.\*gimp.\*RC3" | awk '{print $2}')

This awk command simply prints the second field (which is the package name) of each matching line. It's much cleaner and less prone to errors.

**3. Using cut (Alternative):**

cut can also be used, though awk is usually preferred for more complex scenarios.

Bash

pacman -S $(pacman -Ss | grep -i ".\*ucrt.\*gimp.\*RC3" | cut -d ' ' -f 2)

This cuts the line at spaces (-d ' ') and extracts the second field (-f 2).

**4. Best Practice: -s (sync) vs. -S (sync and install):**

If you're just trying to *find* the package and then potentially install it, it's safer to separate the search and installation steps:

Bash

package=$(pacman -Ss | grep -i ".\*ucrt.\*gimp.\*RC3" | awk '{print $2}')

if [[ -n "$package" ]]; then # Check if a package was found

pacman -S "$package"

else

echo "No matching package found."

fi

This approach avoids trying to install a non-existent package if the grep or awk commands don't find a match. The if statement checks if the package variable is empty.

**Explanation of the Original Error:**

Your original sed command was:

Bash

sed 's/^ucrt64\///;s/ \[installed\]$//'

This removes "ucrt64/" from the beginning and " [installed]" from the end. However, it *doesn't* remove the version number (e.g., "3.0.0RC3-1"). So, if the output of pacman -Ss | grep ... was:

ucrt64/mingw-w64-ucrt-x86\_64-gimp 3.0.0RC3-1 [installed]

Your sed command would transform it into:

mingw-w64-ucrt-x86\_64-gimp 3.0.0RC3-1

Then, pacman -S would try to install *two* packages: "mingw-w64-ucrt-x86\_64-gimp" and "3.0.0RC3-1". Since "3.0.0RC3-1" is not a valid package name, you get the "target not found" error.

By using the corrected sed command, awk, or cut as shown above, you ensure that only the package name is passed to pacman -S, avoiding the error. The best practice of separating the search and install steps adds robustness to your script.