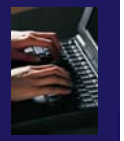
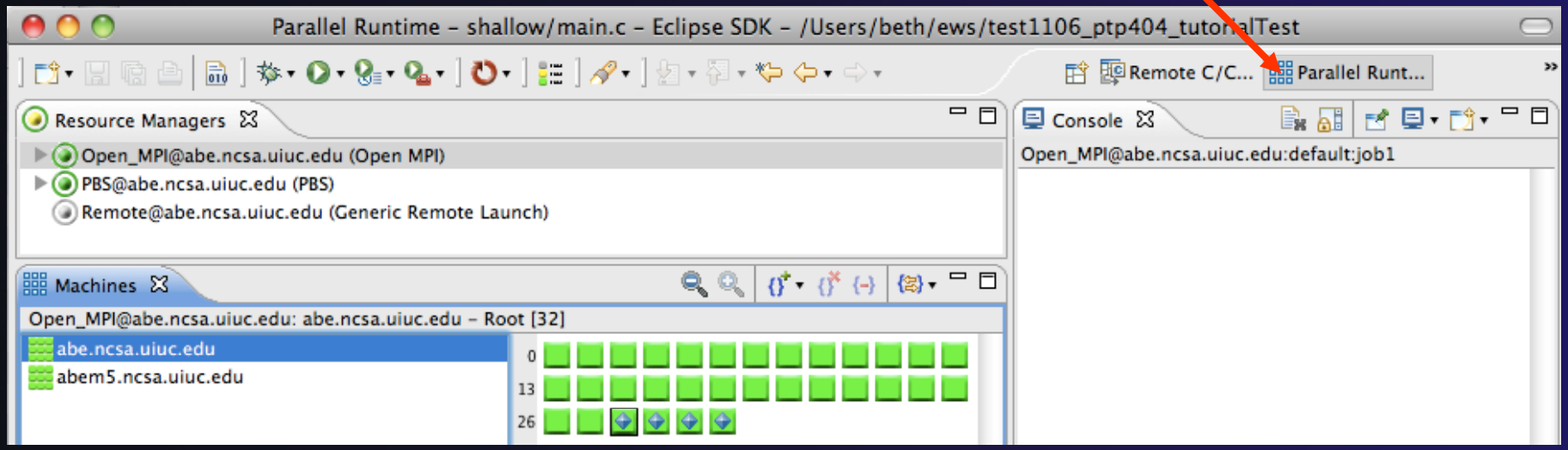
Module 5: Parallel Debugging

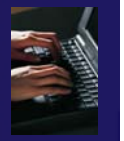
- Objective
  - Learn the basics of debugging parallel programs
- Contents
  - Launching a debug session
  - The Parallel Debug Perspective
  - Controlling sets of processes
  - Controlling individual processes
  - Parallel Breakpoints
  - Terminating processes



Debugging an Application

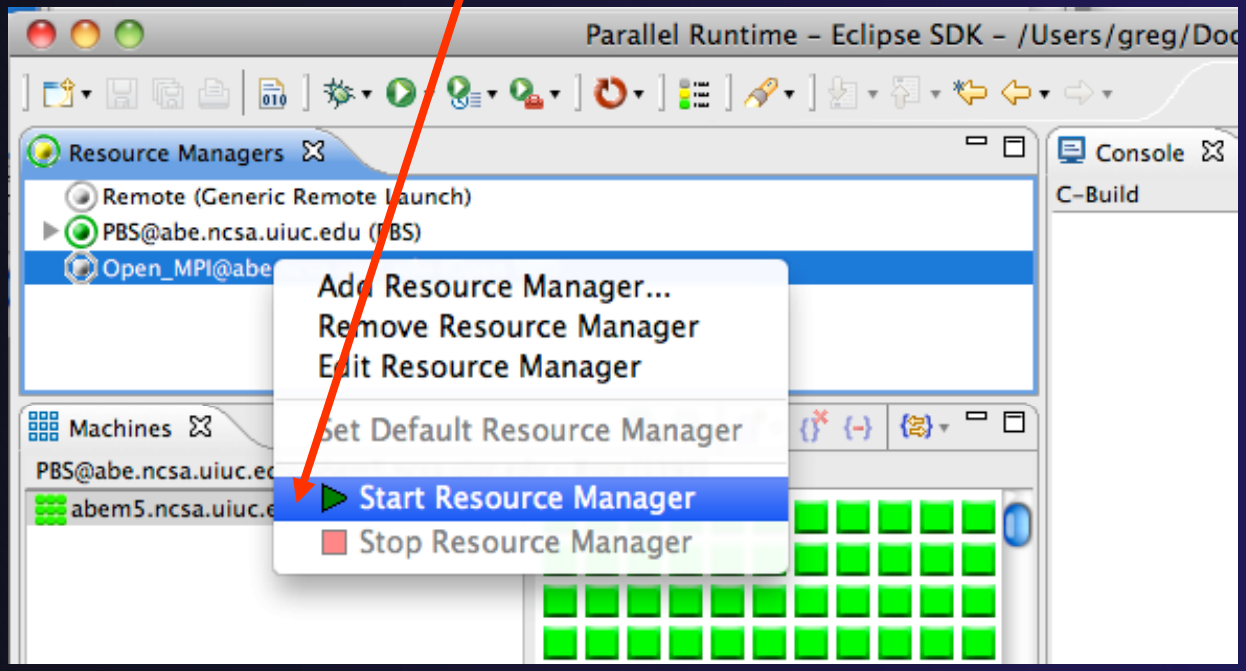
- Debugging requires interactive access to the application
- Since PBS is for batch execution, we will use Open MPI to provide interactive access to the machine (PBS will support interactive execution in the future)
- First switch to the Parallel Runtime perspective if not already there

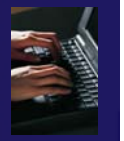




Start the Resource Manager

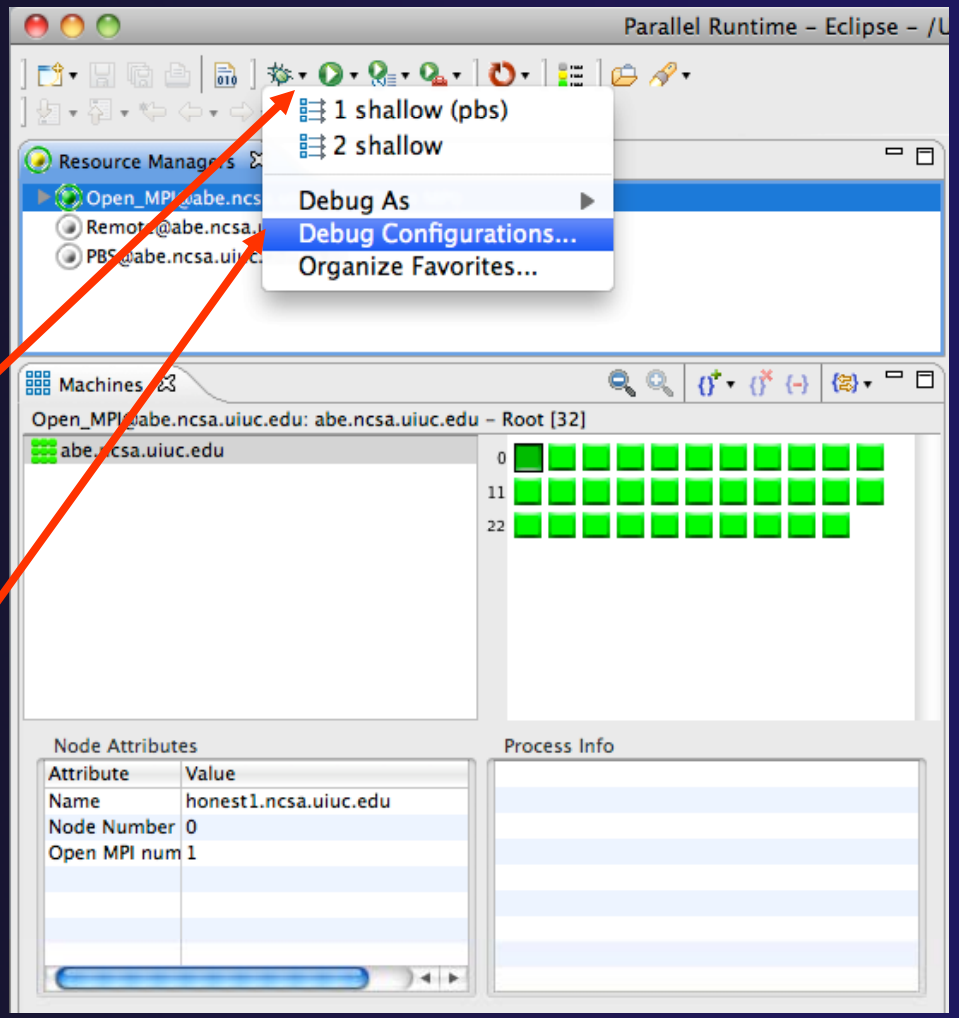
- If the Open_MPI Resource manager is not already started (green icon), start it now:
  - Right-click on the resource manager and select **Start Resource Manager** from the menu

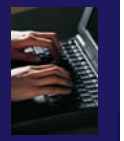




Create a Debug Configuration

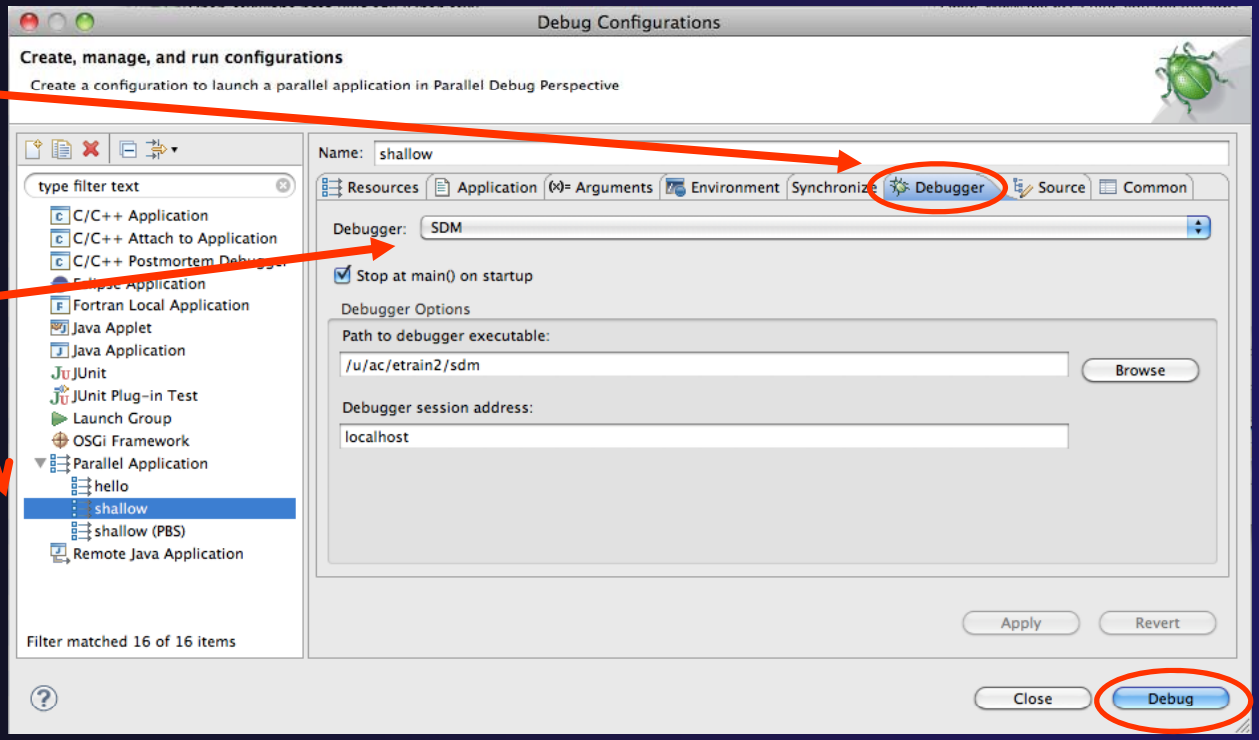
- A debug configuration is essentially the same as a run configuration (like we used in modules 3 & 4)
- We will re-use the existing configuration and add debug information
- Use the drop-down next to the debug button (bug icon) instead of run button
- Select **Debug Configurations...** to open the **Debug Configurations** dialog





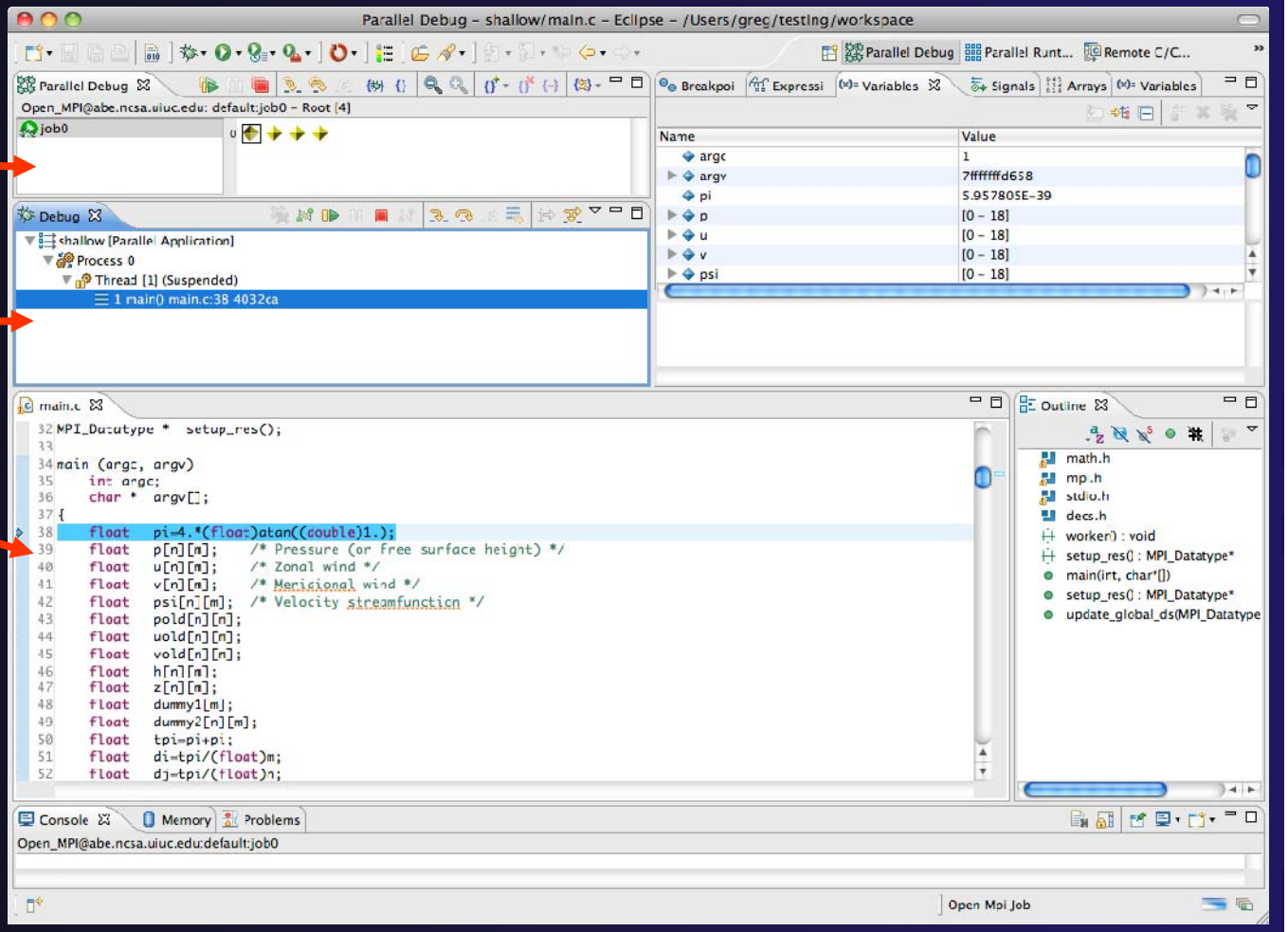
Configure the Debugger Tab

- Select **Debugger** tab
- Select the **shallow** configuration
- Make sure **SDM** is selected in the **Debugger** dropdown
- Check the debugger path is correct
  - Should be the path to the sdm executable on the remote system
- Debugger session address should not need to be changed
- Click on **Debug** to launch the program



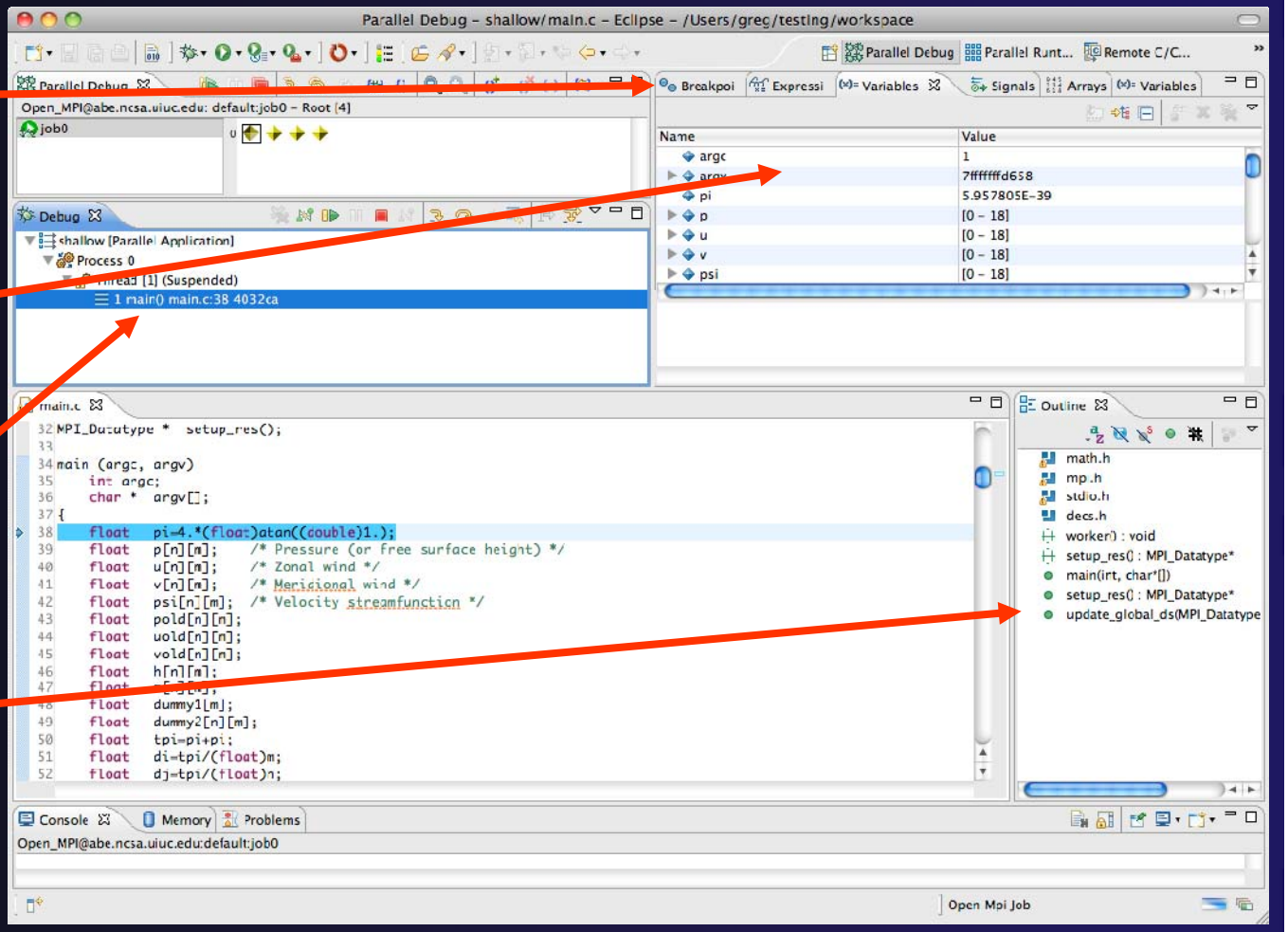
The Parallel Debug Perspective (1)

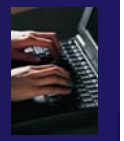
- **Parallel Debug view** shows job and processes being debugged
- **Debug** view shows threads and call stack for individual processes
- **Source** view shows a **current line marker** for all processes



The Parallel Debug Perspective (2)

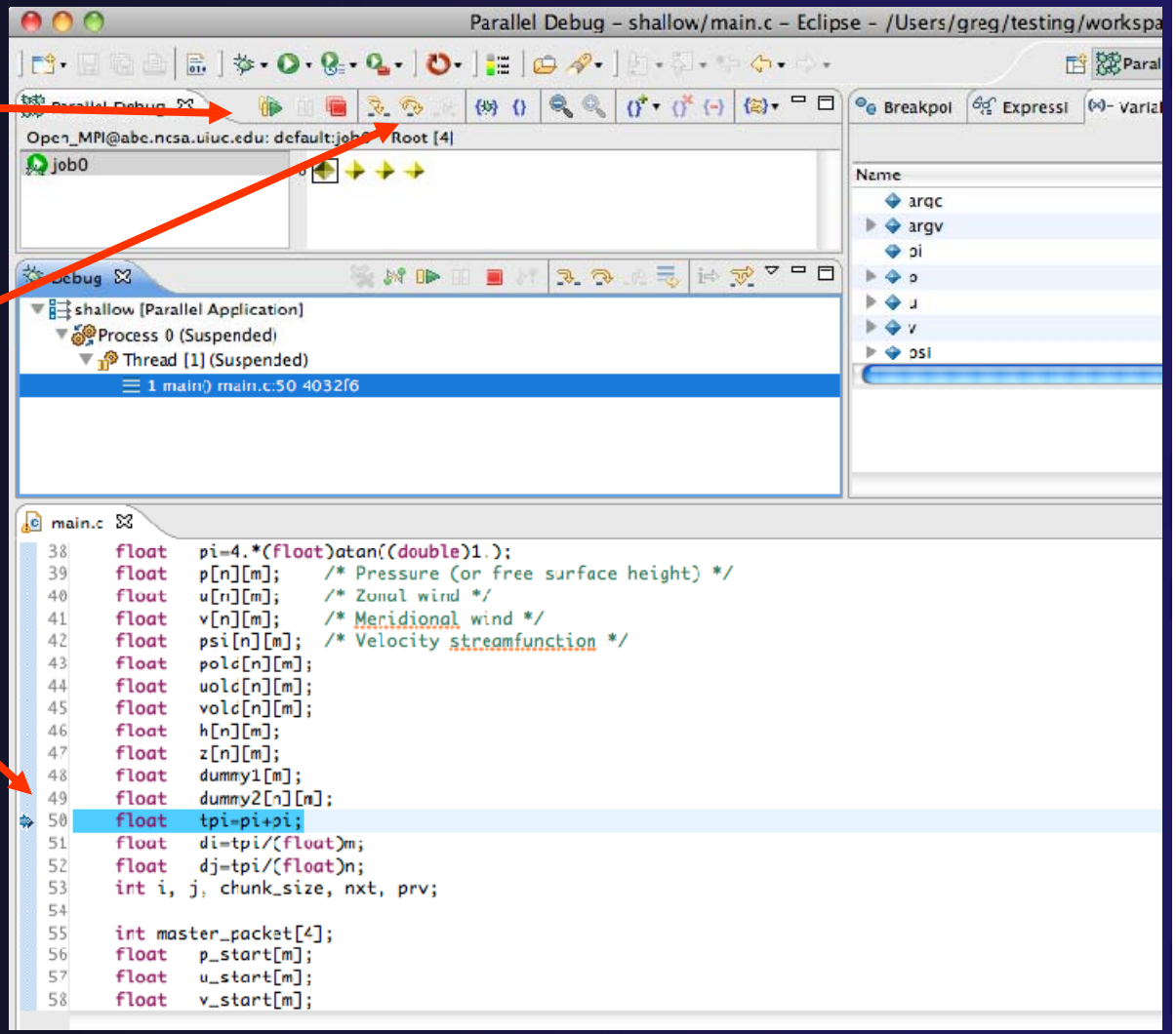
- **Breakpoints** view shows breakpoints that have been set (more on this later)
- **Variables** view shows the current values of variables for the currently selected process in the **Debug** view
- **Outline** view (from CDT) of source code

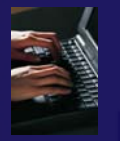




Stepping All Processes

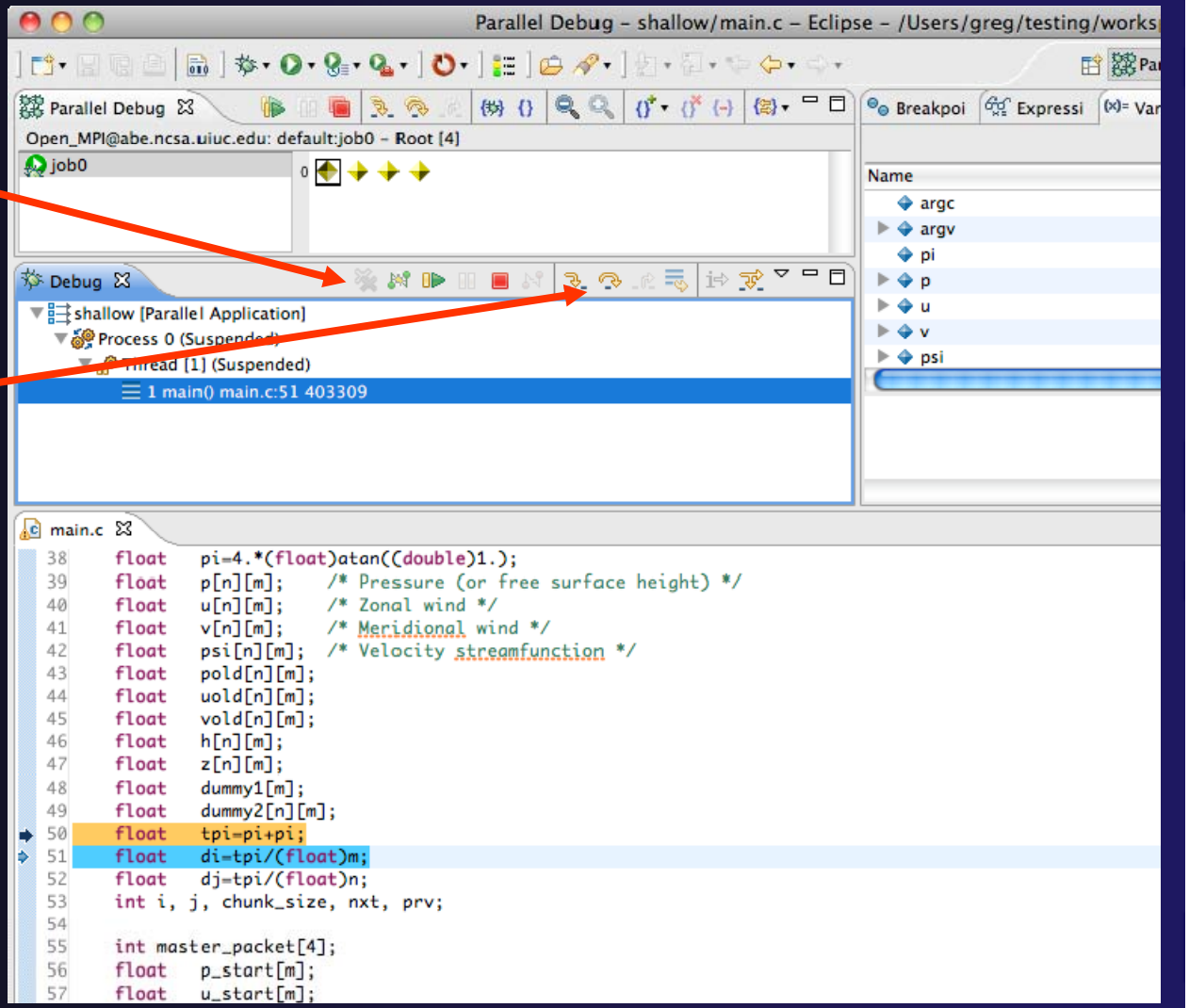
- The buttons in the **Parallel Debug View** control groups of processes
- Click on the **Step Over** button
- Observe that all process icons change to green, then back to yellow
- Notice that the current line marker has moved to the next source line





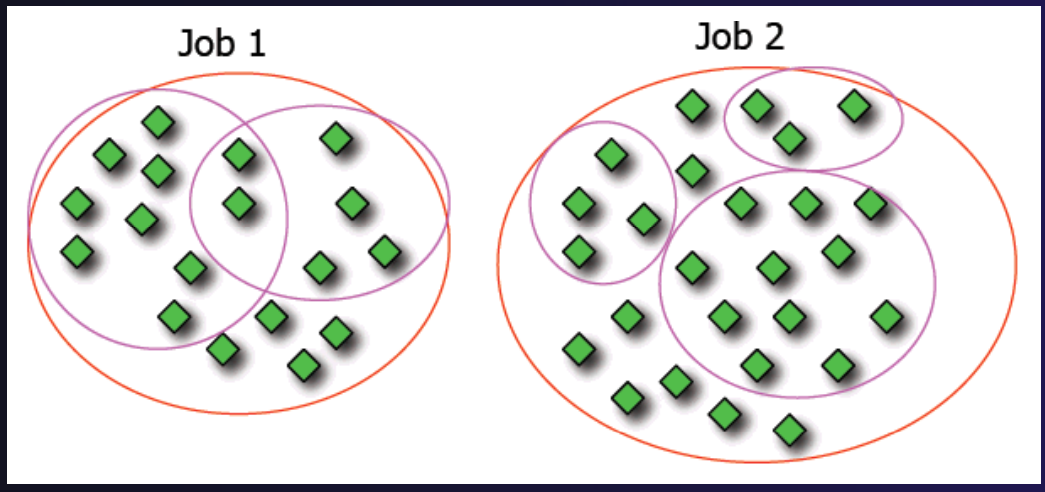
Stepping An Individual Process

- The buttons in the **Debug view** are used to control an individual process, in this case process 0
- Click the **Step Over** button
- You will now see two current line markers, the first shows the position of process 0, the second shows the positions of processes 1-3



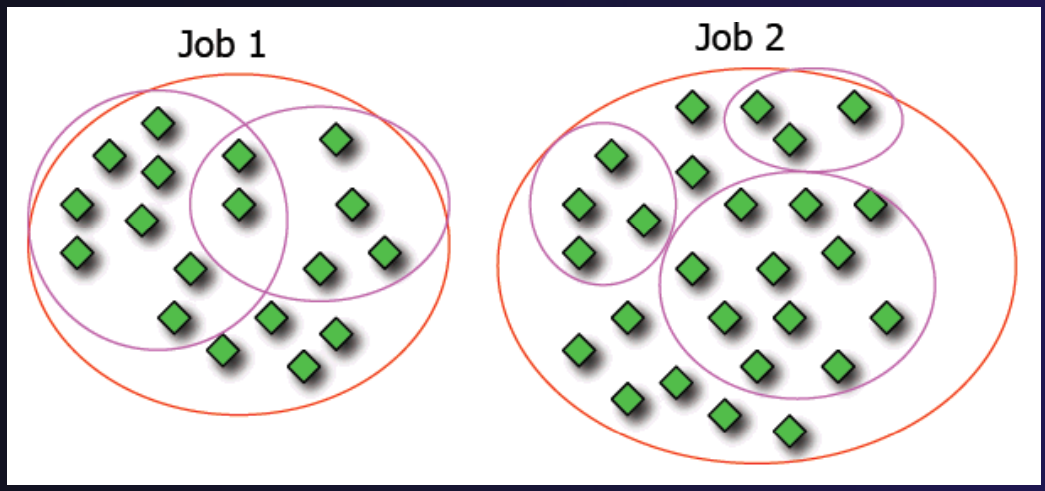
Process Sets (1)

- Traditional debuggers apply operations to a single process
- Parallel debugging operations apply to a single process or to arbitrary collections of processes
- A process set is a means of simultaneously referring to one or more processes



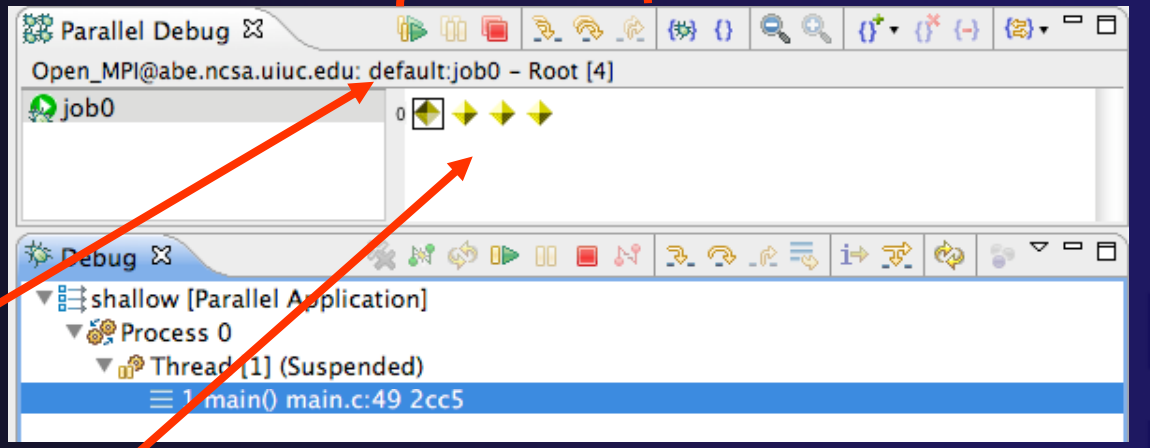
Process Sets (2)

- When a parallel debug session is first started, all processes are placed in a set, called the **Root** set
- Sets are always associated with a single job
- A job can have any number of process sets
- A set can contain from 1 to the number of processes in a job



Operations On Process Sets

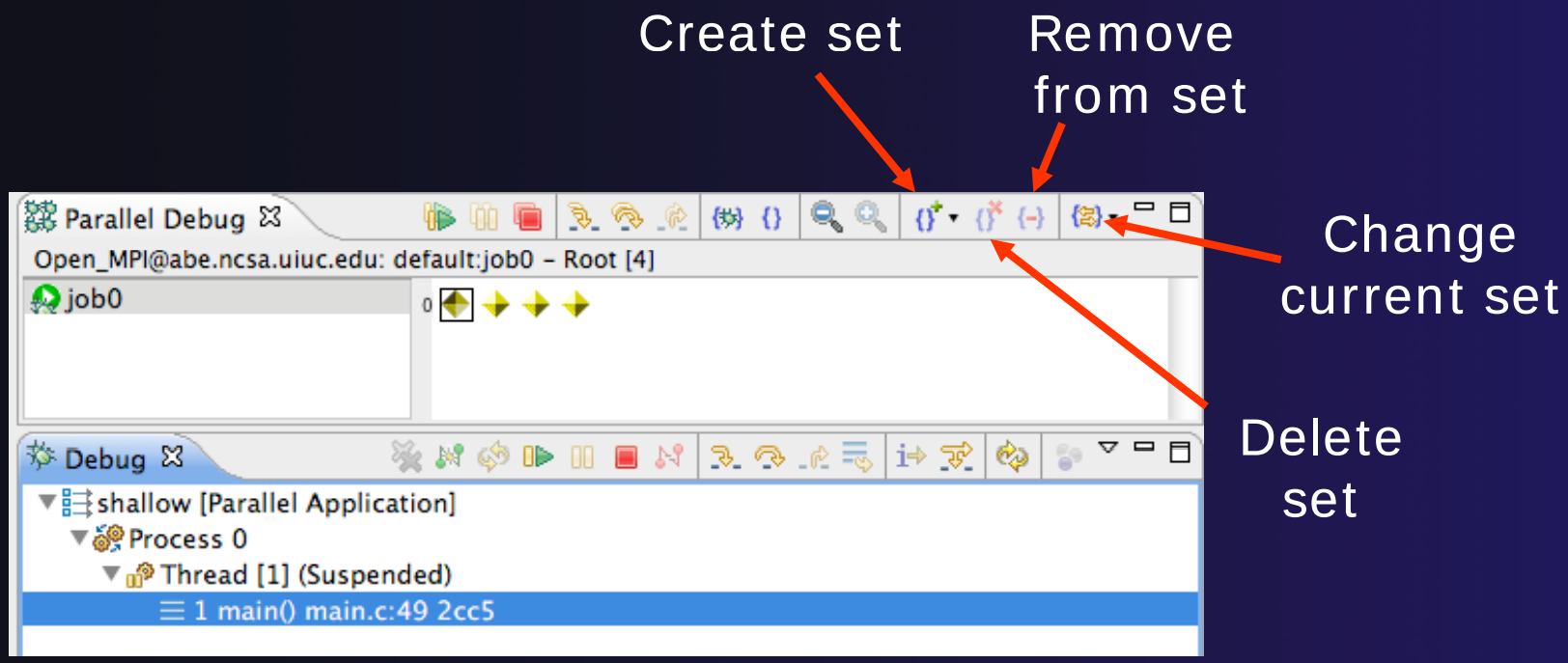
- Debug operations on the **Parallel Debug view** toolbar always apply to the current set:
  - Resume, suspend, stop, step into, step over, step return
- The current process set is listed next to job name along with number of processes in the set
- The processes in process set are visible in right hand part of the view

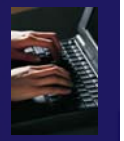


Root set = all processes

Managing Process Sets

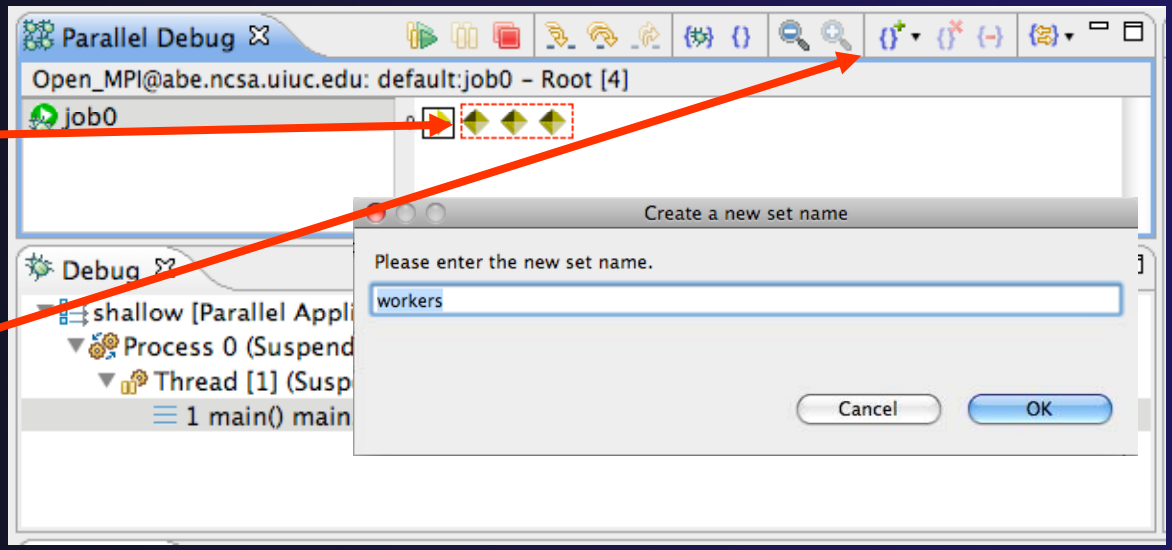
- The remaining icons in the toolbar of the **Parallel Debug view** allow you to create, modify, and delete process sets, and to change the current process set

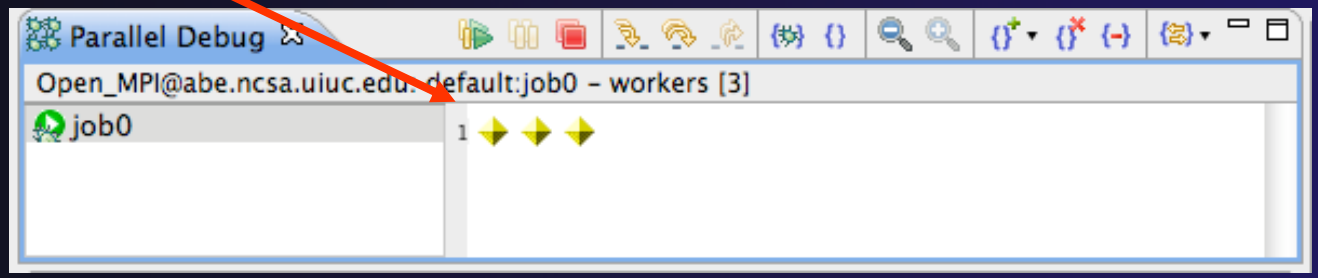


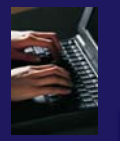


Creating A New Process Set

- Select the processes you want in the set by clicking and dragging, in this case, the last three
- Click on the **Create Set** button
- Enter a name for the set, in this case **workers**, and click **OK**
- You will see the view change to display only the selected processes

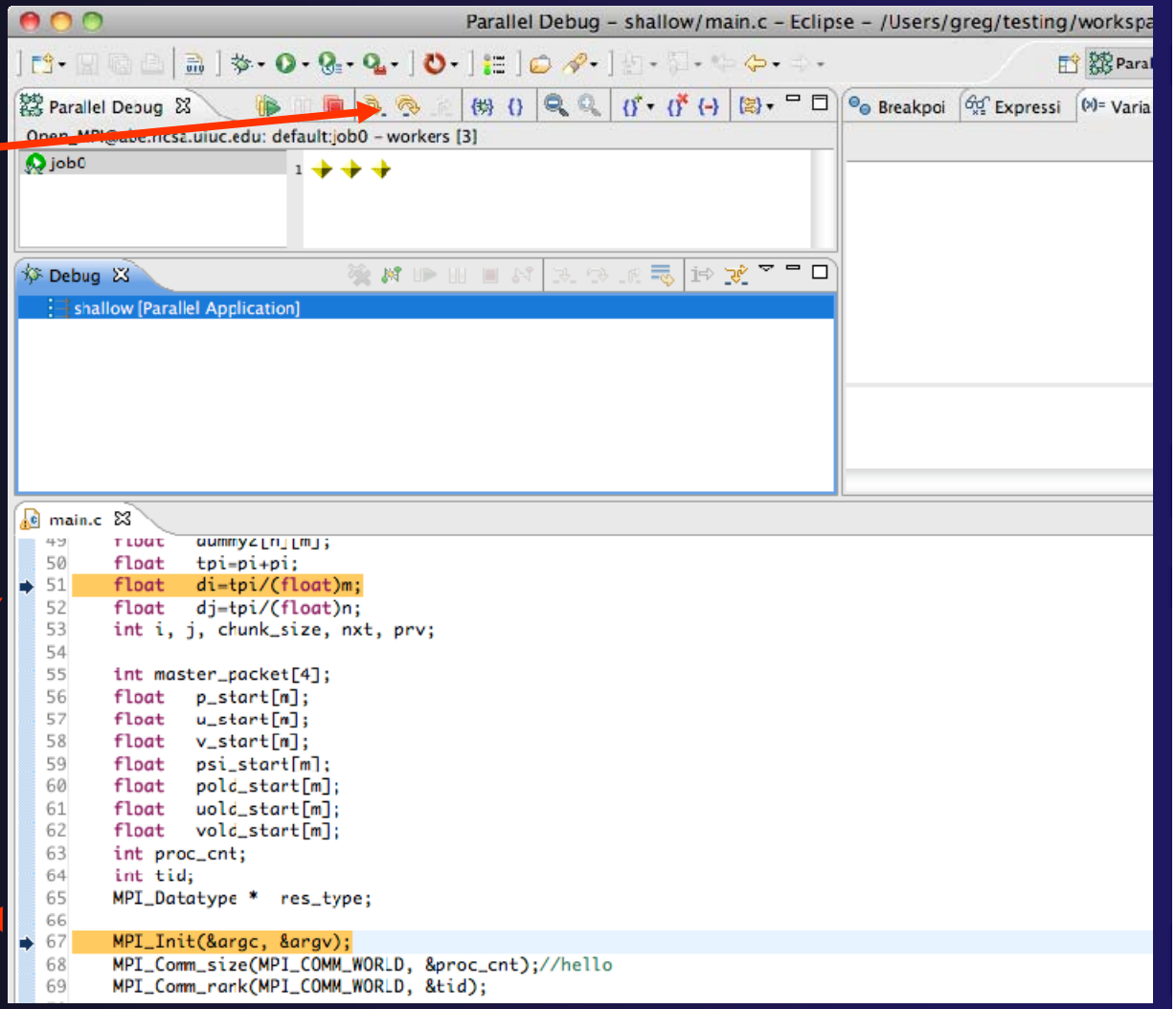






Stepping Using New Process Set

- With the **workers** set active, click the **Step Over** button
- You will see only the first current line marker move
- Step a couple more times
- You should see two line markers, one for the single master process, and one for the 3 worker processes

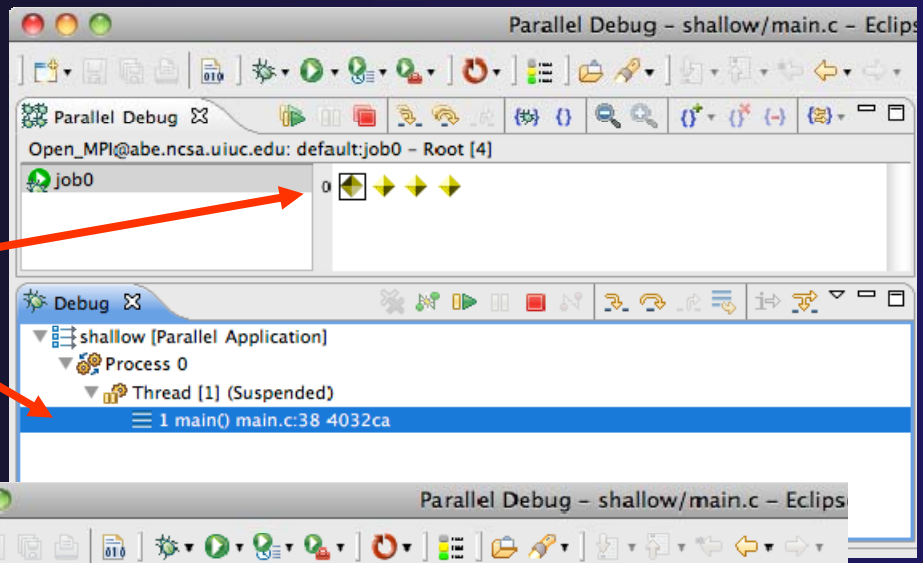


Process Registration

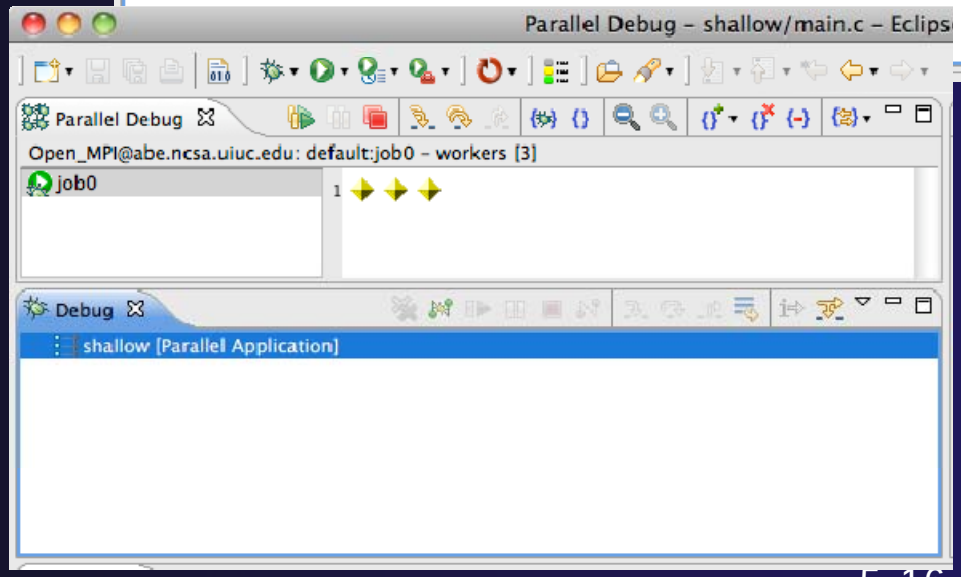
- Process set commands apply to groups of processes
- For finer control and more detailed information, a process can be registered and isolated in the **Debug view**
- Registered processes, including their stack traces and threads, appear in the **Debug view**
- Any number of processes can be registered, and processes can be registered or un-registered at any time

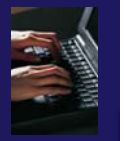
Process Registration (2)

- By default, process 0 was registered when the debug session was launched
- Registered processes are surrounded by a box and shown in the Debug view



- The Debug view only shows registered processes in the current set
- Since the “workers” set doesn’t include process 0, it is no longer displayed in the Debug view

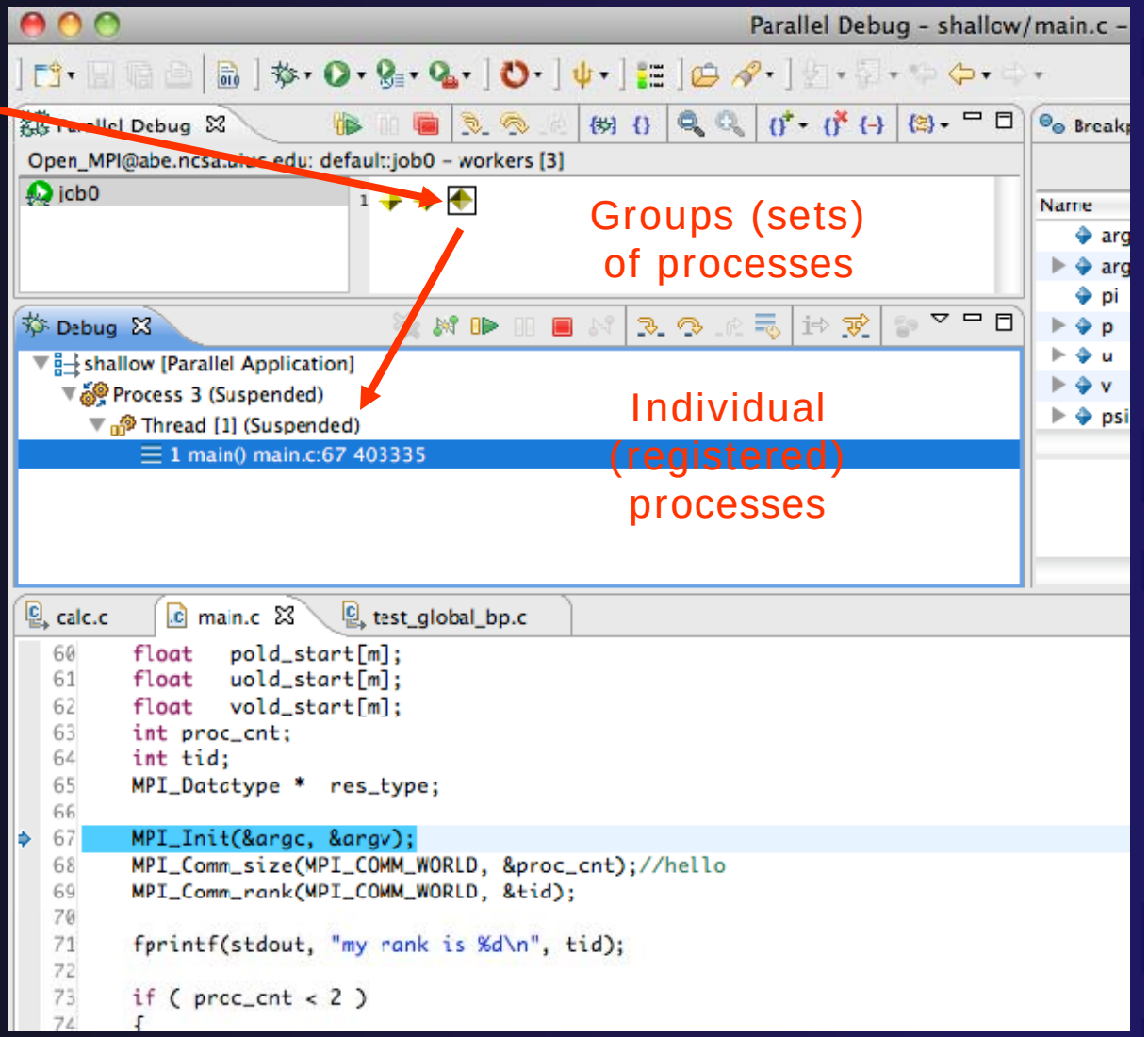




Registering A Process

- To register a process, double-click its process icon in the **Parallel Debug view** or select a number of processes and click on the **register** button

- To un-register a process, double-click on the process icon or select a number of processes and click on the **unregister** button

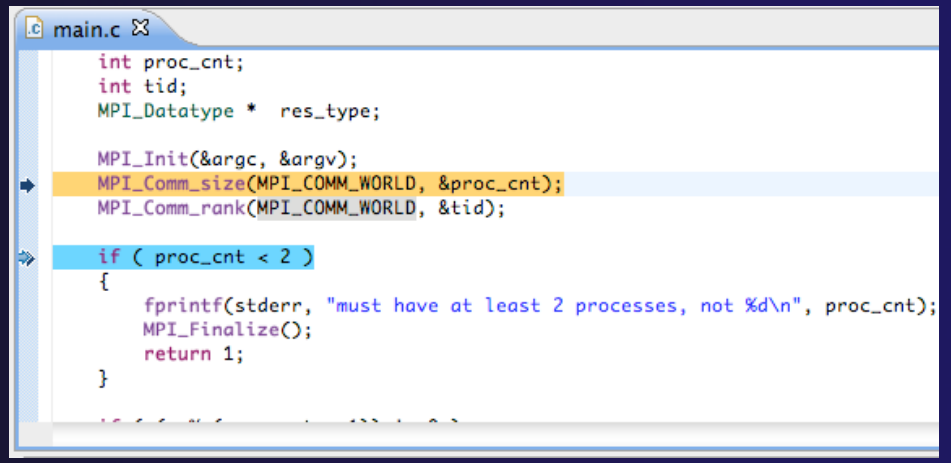


Current Line Marker

- The current line marker is used to show the current location of suspended processes
- In traditional programs, there is a single current line marker (the exception to this is multi-threaded programs)
- In parallel programs, there is a current line marker for every process
- The PTP debugger shows one current line marker for every group of processes at the same location

Colors And Markers

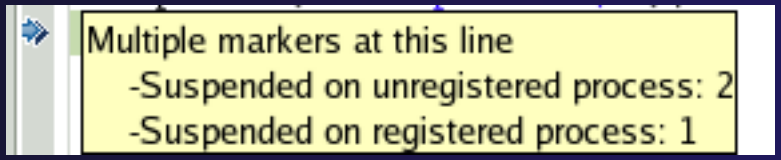
- The highlight color depends on the processes suspended at that line:
  - **Blue:** All registered process(es)
  - **Orange:** All unregistered process(es)
  - **Green:** Registered or unregistered process with no source line (e.g. suspended in a library routine)
- The marker depends on the type of process stopped at that location
- Hover over marker for more details about the processes suspend at that location



 Multiple processes marker

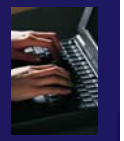
 Registered process marker

 Un-registered process marker



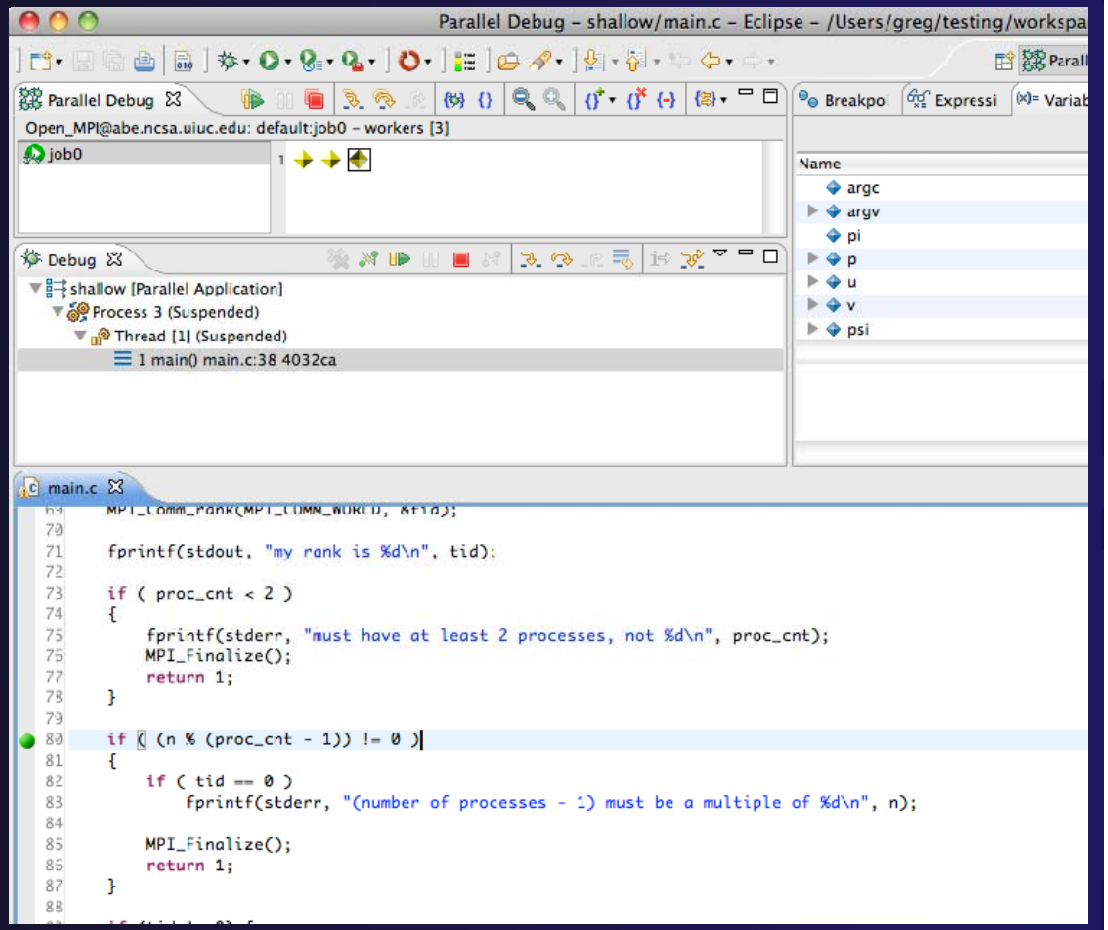
Breakpoints

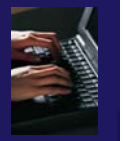
- Apply only to processes in the particular set that is active in the **Parallel Debug view** when the breakpoint is created
- Breakpoints are colored depending on the active process set and the set the breakpoint applies to:
  - Green indicates the breakpoint set is the same as the active set.
  - Blue indicates some processes in the breakpoint set are also in the active set (i.e. the process sets overlap)
  - Yellow indicates the breakpoint set is different from the active set (i.e. the process sets are disjoint)
- When the job completes, the breakpoints are automatically removed



Creating A Breakpoint

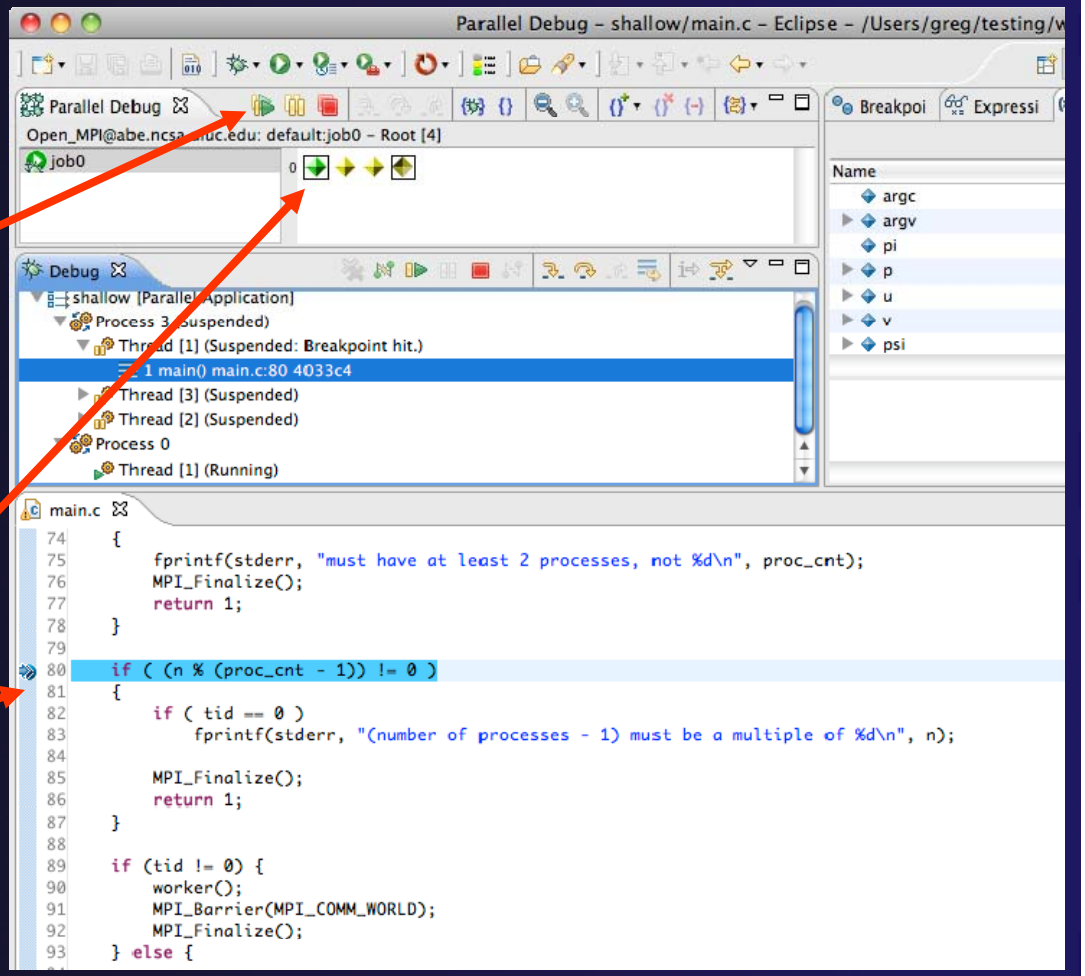
- Select the process set that the breakpoint should apply to, in this case, the **workers** set
- Double-click on the left edge of an editor window, at the line on which you want to set the breakpoint, or right click and use the **Parallel Breakpoint ▶ Toggle Breakpoint** context menu
- The breakpoint is displayed on the marker bar

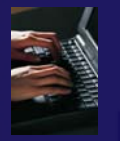




Hitting the Breakpoint

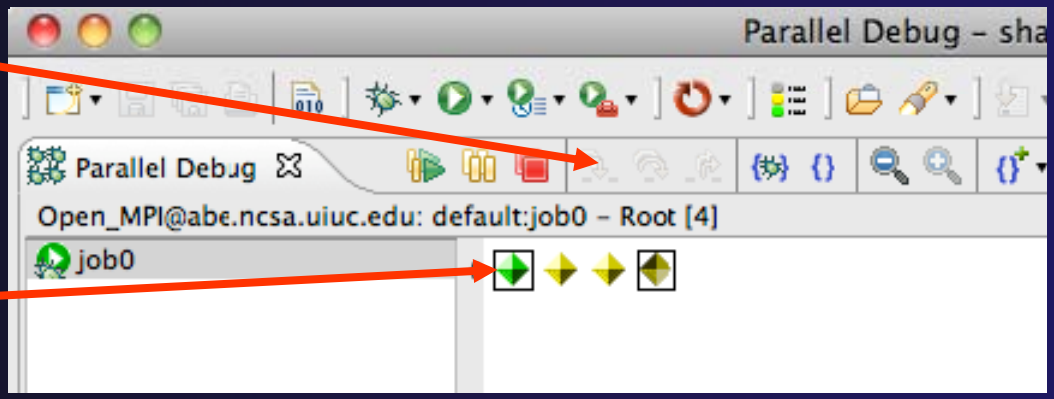
- Switch back to the **Root** set by clicking on the **Change Set** button
- Click on the **Resume** button in the **Parallel Debug view**
- In this example, the three worker processes have hit the breakpoint, as indicated by the yellow process icons and the current line marker
- Process 0 is still running as its icon is green
- Processes 1-3 are suspended on the breakpoint



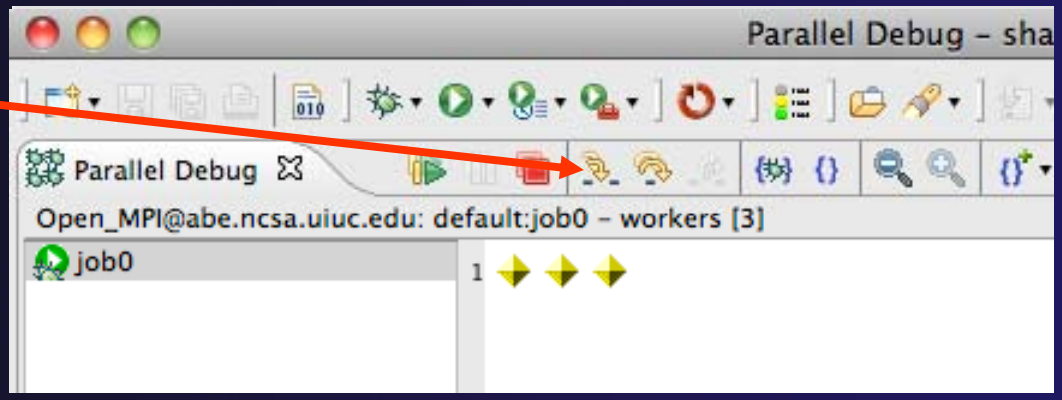


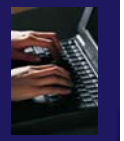
More On Stepping

- The **Step** buttons are only enabled when all processes in the active set are **suspended** (yellow icon)
- In this case, process 0 is still running



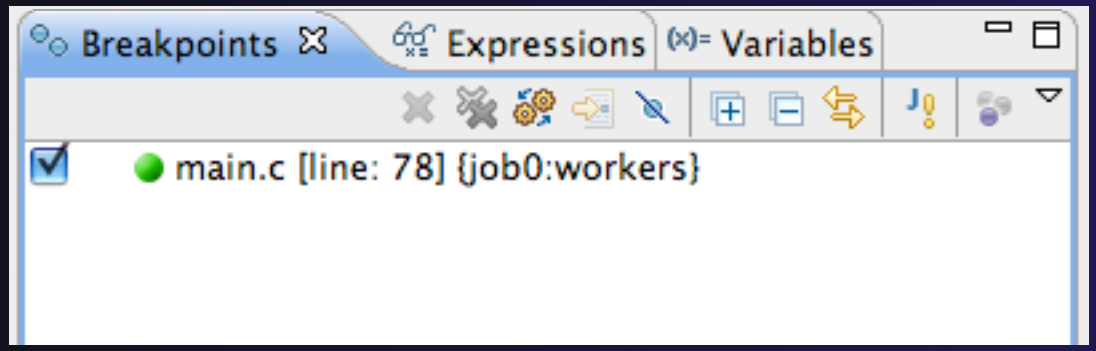
- Switch to the set of suspended processes (the **workers** set)
- You will now see the **Step** buttons become enabled

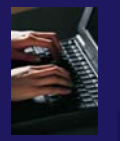




Breakpoint Information

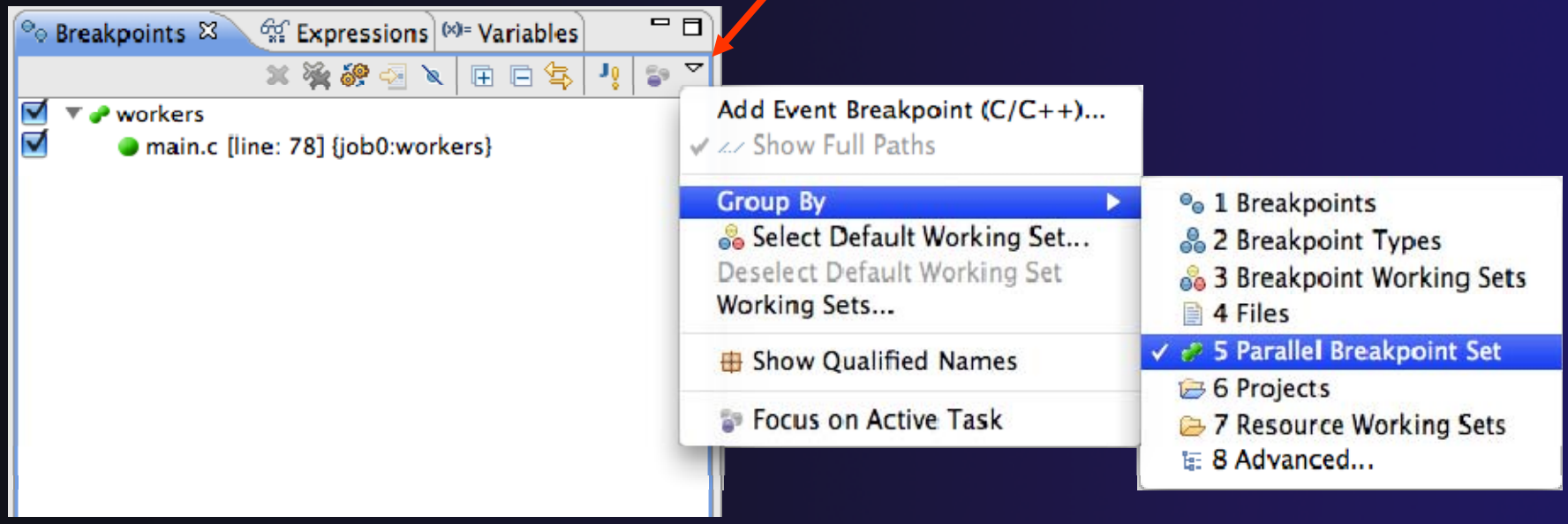
- Hover over breakpoint icon
  - Will show the sets this breakpoint applies to
- Select **Breakpoints** view
  - Will show all breakpoints in all projects





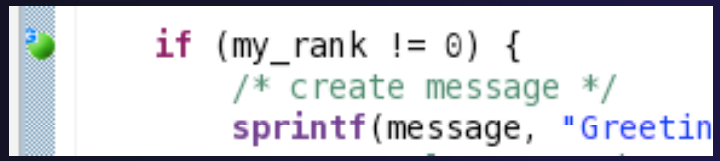
Breakpoints View

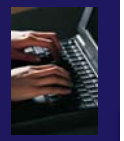
- Use the menu in the breakpoints view to group breakpoints by type
- Breakpoints sorted by breakpoint set (process set)



Global Breakpoints

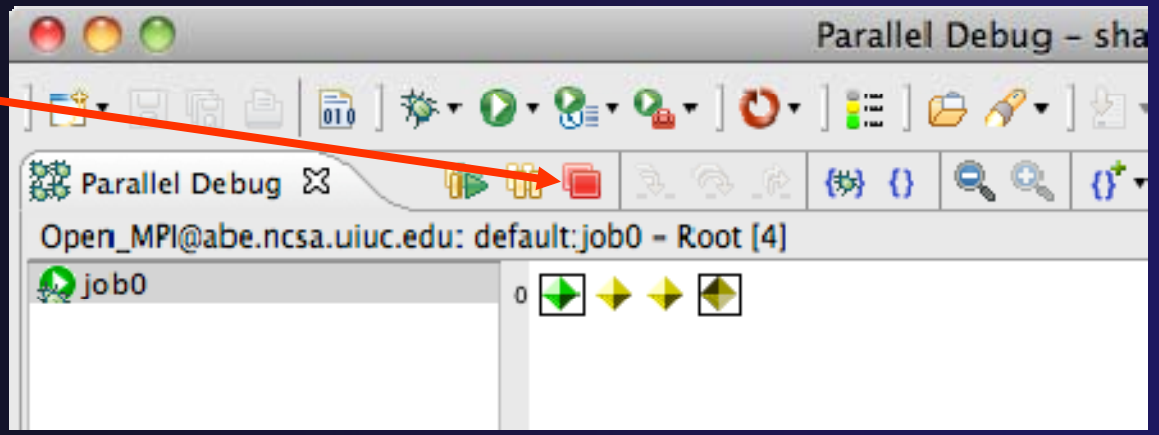
- Apply to all processes and all jobs
- Used for gaining control at debugger startup
- To create a global breakpoint
  - First make sure that no jobs are selected (click in white part of jobs view if necessary)
  - Double-click on the left edge of an editor window
  - Note that if a job is selected, the breakpoint will apply to the current set





Terminating A Debug Session

- Click on the **Terminate** icon in the **Parallel Debug view** to terminate all processes in the active set
- Make sure the **Root** set is active if you want to terminate all processes



- You can also use the terminate icon in the **Debug** view to terminate the currently selected process